

Express Mail No. EL727968344US

Date of Deposit: June 1, 2001

APPLICATION FOR LETTERS PATENT
OF THE UNITED STATES

NAME OF INVENTORS: TUAN NGUYEN

730 Henley Court
Duluth, Georgia 20097
USA

ANTHONY PETER DUCK

5326 Tortuga Trail
Austin, Texas 78731
USA

IAN RAWLES

495 Persimmon Lane
Roswell, Georgia 30076
USA

THOMAS MAIR

Lindenalle 27
82061 Neuried
Germany

ROBERT GRAY

5730 Lake Windsor Pky.
Buford, Georgia 30518
USA

TITLE OF INVENTION: ELECTRONICS ASSEMBLY SYSTEMS
CUSTOMER BENEFIT MODELING
TOOLS AND METHODS

TO WHOM IT MAY CONCERN, THE FOLLOWING IS
A SPECIFICATION OF THE AFORESAID INVENTION

09873021.060101

ELECTRONICS ASSEMBLY SYSTEMS CUSTOMER BENEFIT MODELING TOOLS AND METHODS

CROSS-REFERENCE TO RELATED APPLICATIONS

This applications claims the benefit for purposes of priority, under 35 U.S.C. §119(e), of U.S. Patent Application Serial Nos. 60/208,664, filed June 1, 2000, and 60/280,679 filed March 30, 2001, the contents of which are herein incorporated by reference in their entirety.

FIELD OF THE INVENTION

The present invention relates in general to the field of electronics assembly systems and, in particular, to tools and methods for customer benefit modeling for purposes of electronics assembly system configurations.

BACKGROUND OF THE INVENTION

Numerous methods and tools exist for modeling and simulating production and assembly lines. For example, a spreadsheet such as Microsoft Excel® 98 can be used to perform static modeling of an assembly line. While spreadsheets are easy to use, and may provide results based on static modeling, they do not adequately take into account how various factors interact with each other. For instance, a spreadsheet can be configured to show that certain pieces of equipment in an assembly line may have a downtime of 25%, but it cannot be configured adequately to show that at various times, all of these pieces of equipment might be down at the same time, much less predict the actual impact such an event might have on line performance.

Discrete event simulation software products, such as Witness® 2000 (available from Lanner Group, Greenwich England), provide better estimates of how a line will actually operate. One drawback of using existing discrete event simulation products is the amount of time and skill it takes to set up a discrete event simulation of an actual or proposed system. Most discrete event simulation packages require that highly detailed simulation objects be created by the person performing the modeling. These objects contain

numerous parameters, some of which may be variable in nature and some of which are inherent in the piece of equipment that the simulation object represents.

Consultants typically spend a great deal of time, sometimes several weeks, building a discrete event simulation model. This long turn-around time is not helpful for a salesperson pitching equipment, or consultants trying to demonstrate their ability to increase a line's efficiency. It would be preferable if a consultant or a salesperson who is attempting to sell products or services could deliver a persuasive, well-supported sales pitch, or needed consultation in far less time.

Prior to the present invention, there appears to have been no adequate way for a consultant to quickly and easily build and run a simulation model for a proposed line configuration. Moreover, the level of skill required to build a simulation often exceeds that possessed by a typical salesperson. It would therefore be valuable to have tools and methods that: demand only a low level of skill to use, similar to that required to use a spreadsheet; can be quickly programmed; and give accurate results, similar to those of a sophisticated discrete event simulation modeling tool.

SUMMARY OF THE INVENTION

The present invention provides methods and tools for modeling a configuration for, among other things, an electronics assembly system and generating performance measures associated with a given configuration based upon a simulation of the behavior of the system. In particular, the methods and tools make it possible for electronics assembly equipment salespersons or consultants to generate a simulation model for purposes of a sales pitch or a consulting project on the order of magnitude of about a half an hour, or even less, to conduct model-based prediction of system performance under various configuration assumptions, and to generate for an actual or prospective customer one or more expected cost of ownership values for particular configuration assumptions.

The customer benefit modeling tool of the present invention enables trained consultants to accurately model current and future customer assembly systems at the line, multi-line, shop floor and factory levels. It also enables the performance of customer systems to be expressed in a range of key metrics that include, for example, throughput, cost performance, yield, OEE and COO. The customer benefit tool according to the present invention can be used to accurately represent production reality because, in one embodiment, it can handle distributed inputs and events by using a dynamic model based on discrete event simulation. This modeling approach enables separate analysis of system design issues (e.g. equipment configuration, wip buffer sizing) and operations issues (e.g. staffing, material availability).

The customer benefit modeling tool can also provide customer/project specific target benchmarking under defined, ideal operating conditions (e.g. no break downs, no run outs). That is, the performance that one would expect under short duration "acceptance testing." The tool may also be configured to enable target production metrics to be established based on realistic conditions. It can also be configured to model the impact of changing operations related variables.

The tool advantageously allows customer models to be validated under "acceptance test" conditions to ensure that the machine based processes and cycle times have been accurately modeled. As part of the validation process, the tool may allow a customer-specific model to be calibrated against real data. The validation process may focus on throughput or yield. In one embodiment, throughput is dominant factor.

In accordance with the present invention, a user proposes a configuration for an assembly line by selecting from among objects that represent assembly line equipment and parameters characterizing the operation of the equipment. The assembly line equipment often has various operating characteristics well-known to industrial engineers (e.g., mean time between failure, mean time to repair, how often a piece of equipment needs manual assistance, how long the manual assistance takes, etc.). Typically, these operating characteristics are empirically derived statistical measures or

models that may be fixed or that may vary as a function of one or more variables. Sometimes, the order in which the equipment will be laid out to form a line may also be part of the configuration process. Specific values for the operating characteristics, determined statistically, are associated with each piece of equipment in the proposed line configuration. Shift data, production part data, and other line parameters may also be assigned values. The configuration and associated operating characteristic values are then used to build a discrete event simulation.

In some embodiments of the present invention, building a discrete event simulation requires creating detailed simulation objects. To streamline the building of a simulation by selecting and arranging the simulation objects, templates may be created in advance for each simulation object and values may be read into the template to create the simulation object. After the discrete event simulation is run through a suitable number of trials, performance results gathered for those trials are exported to a reporting means. The reporting means may contain an interface that allows the user to input specific data, which in turn would allow for custom reports to be generated that present the performance results in a manner that can be tailored by a salesperson, customer, or consultant to the needs of the customer or the situation.

The assembly equipment modeling and simulation according to the present invention may be performed virtually anywhere, using suitable computer software, further described below, that runs solely on the process of any type of personal computer, or in a client-server fashion over a network, such as the Internet, or in communication with sites on the World-Wide-Web.

The computer software provides a means for representing assembly line components and parameters in a configuration that is chosen on the fly. A means for reading data and parameters about each selected component and proposed line configuration and parameter setting is also provided. A simulation generating means takes the inputted line configuration and parameter setting and associated data and uses that information to generate a simulation, which in some embodiments involves building simulation objects

from templates. Output information derived by running the simulation over a suitable number of trials is sent to a reporting means.

The means for representing the assembly line components in the user-chosen configuration and having user-chosen parameter settings, as well as the reporting means, may take the form of a spreadsheet. Since the reporting means may be a spreadsheet, it may be configured to accept user-specific inputs and generate custom reports based on this information.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a flow chart illustrating the overall file structure and logic flow of one embodiment of the present invention.

Figures 2-2A are screen views of spreadsheets (e.g., in MS Excel®) that are configured to act as an input interface in an embodiment of the present invention.

Figure 3 illustrates the files and directories that are set up when the tool according to an embodiment of the present invention is run.

Figures 4-4C are screen views of spreadsheets (e.g., in MS Excel®) configured to act as interface for inputting shift schedule data in an embodiment of the present invention.

Figures 5-5C are screen views of spreadsheets (e.g., in MS Excel®) configured to act as an interface for inputting line configurations in an embodiment of the present invention.

Figure 6 illustrates how pull-down menus, in an embodiment of the present invention, may be used with the screen shown in Figure 5.

Figures 7-7F illustrates a partially populated line configuration spreadsheet interface in an embodiment of the present invention.

Figures 8-8B illustrates a board data input interface in an embodiment of the present invention.

Figures 9-9B illustrates a built simulation, in an embodiment of the present invention, that is ready to be run.

09873021-060101

Figure 10 is a screen view of an output interface in the form of a spreadsheet that is capable of generating customer specific reports, in an embodiment of the present invention.

Figure 11 is a screen view of the screen that appears, in an embodiment of the present invention, when the "Boundaries" button of Figure 10 is selected.

Figure 12 is a screen view of the screen that appears, in an embodiment of the present invention, when the "Set-up" button of Figure 10 is selected.

Figure 13 is a screen view of the screen that appears, in an embodiment of the present invention, when the "Quality" button of Figure 10 is selected.

Figure 14 is a screen view of the screen that appears, in an embodiment of the present invention, when the "Maintenance" button of Figure 10 is selected.

Figure 15 is a screen view of the screen that appears, in an embodiment of the present invention, when the "Cost of Ownership" button of Figure 10 is selected.

Figure 16 is a screen view of a graphical report of Cost of Ownership, in an embodiment of the present invention.

Figure 16A is a flow-chart illustrating a cost of ownership routine, in an embodiment of the present invention.

Figure 17 is a flow-chart illustrating a process for using the Customer Benefit Tool, in an embodiment of the present invention.

Figure 18 is a diagram showing a set of relationships between customer benefit modeling according to the present invention to various aspects of objectives of a business model that may employ aspects of the present invention.

Figure 19 is a diagram showing a set of relationships between customer benefit modeling according to the present invention to various aspects of objectives of an alternative business model that may employ aspects of the present invention.

Figure 20 illustrates an electronics assembly systems solutions business model embodying and employing aspects of the present invention.

Figure 21 graphically shows electronics assembly customer benefits and enablers provided by various aspects of the present invention.

Figure 22 graphically illustrates an electronics assembly solutions value proposition from a customer project life cycle view.

Figure 23 is a high-level flow-chart illustrating a consultative sales process business method according to the present invention.

Figure 24 is a flow-chart illustrating a method for using the customer benefit modeling tool of the present invention to develop a basic line scenario.

Figure 25 is a flow-chart illustrating customer project life cycle development in accordance with the present invention..

Figure 26 is a flow-chart illustrating customer project life cycle development with added optimized design capabilities, in accordance with the present invention.

Figure 27 is a flow-chart illustrating customer project life cycle development with added automated performance monitoring capabilities, in accordance with the present invention.

Figure 28 is a flow-chart illustrating customer project life cycle development with added capabilities to share data between design and monitoring activities, in accordance with the present invention.

Figure 29 is a flow-chart illustrating customer project life cycle development with added optimized performance capabilities in which a common database is used, in accordance with the present invention.

Figure 30 is a flowchart illustrating an embodiment of a consultative sales method according to the present invention.

Figure 31 is a flowchart illustrating an alternative to the embodiment of Figure 30, which steps providing for the modification of a proposed assembly system solution.

Figure 32 is a flowchart illustrating an embodiment of a consultative sales method according to the present invention, in which the configuration,

modeling and evaluation of electronics assembly configurations is performed in an iterative manner until a performance measure satisfies one or more customer constraints and an offer is made.

Figure 33 is a flowchart illustrating an alternative embodiment of a consultative sales method according to the present invention, in which the configuration, modeling and evaluation of electronics assembly configurations are performed

Figure 34 is a flowchart illustrating a method for pricing an electronics assembly system solution.

Figure 35 is a flowchart illustrating a method for financing the price of a customer's purchase of an electronic assembly system solution.

Figure 36 is a flowchart illustrating a method for selling a customer an electronic assembly system solution.

DETAILED DESCRIPTION OF THE INVENTION

The present invention provides, among other things, tools and methods that enable quick and easy modeling and simulation of production assembly lines without the need to spend days or weeks building simulation models. It is particularly well-suited for use in modeling electronic component assembly production lines. Of course, it is also useful in modeling other assembly and manufacturing processes. Because the present invention enables quick, efficient modeling that typically can be performed in approximately 30 minutes or less, it is an ideal tool for use by electronics assembly equipment salespersons, as well as consultants and customers.

In accordance with the present invention, a consultant, sales person, customer, or an assembly line designer can quickly perform sophisticated simulations on a computer by inputting or selecting simple designer computer representations, e.g., designer objects, which represent particular pieces of equipment that are used in an assembly line, from a set of such representations created in advance. In the description of the embodiments, "designer objects" and "simulation objects" are described, but the invention contemplates the use of any suitable computer representations. The

computer is configured to take this list of designer objects, and import from a database key parameters for each designer object in the list, as well as other production line data; launch a discrete event simulation tool, which contains templates for detailed simulation objects; build detailed simulation objects from the templates and the designer objects; run a simulation, using simulation software, such as Witness®, ProModel®, Automod®, Simul8®, Arena®, Saimple®; and output specific simulation and modeling parameters in a user friendly manner.

The present invention achieves its efficiency for several reasons. Notably, the populating of detail-intensive simulation objects in the simulation tool is largely automated. A user performing the simulation needs only a low level of computer skill and does not need to spend large amounts of time setting up and building a simulation. In one embodiment, the user input interface takes the form of a spreadsheet, preferably an MS Excel® spreadsheet. From the spreadsheet, the user can build a simulation model in a sophisticated simulation software package, examples of which were listed above, launch the simulation software, and run the simulation. After a particular simulation is run, the outputs may stored in a separate spreadsheet.

Figure 1 provides a high-level logic flow for one embodiment of the present invention. As is illustrated in Figure 1, several different user interfaces are available. MS Excel®, spread sheets may be configured to function as input interfaces. In some embodiments, Visual Basic forms and Web forms may also be configured to function as the input interfaces. Not all of the input interfaces needs to take the same form. For example, a salesperson's interface may be a spread sheet, a customer's may be a web form, and a consultant's may be a Visual Basic form. Those skilled in the art will recognize that various software can be used to generate the input interfaces and in some cases they may be written from scratch by a skilled programmer.

By having multiple interfaces available, the interfaces may be adapted to a specific type of user. For example, there may be one interface for use by

salespersons, another for use by consultants, and yet another for use by customers. To facilitate input, dialog boxes may be created as part of the input interfaces. This allows for easy inputting of data and allows for selecting in advance the number and type of inputs available.

After the user enters the requisite data via a user input interface, step 1 in Figure 1, a routine is run to create and put the data in correct files and formats necessary to run a simulation. The routine at step 2 formats the data that was entered through the interface at step 1 for use with a particular simulation software package according to the requirements of that package. For example, the interface may allow a user to input cycle time in hours or minutes, but the simulation software may need cycle time in seconds. As a further example, Witness® 2000 has a specific format for shift data; however, it may be easier for a user to input shift data in terms of start time, break time, and end time. The routine at step 2 takes the shift data as entered by the user and builds files and structures that can be used by the simulation package (e.g., Witness® 2000 or other available package). In some embodiments, data may be exported from the input interface directly to the simulation package.

The routine in step 2, builds the module that is shown in Step 3 as the CBM_Transfer.XLS module. This module may, but does not necessarily, take the form of a spreadsheet. This module contains all the files and data in the format required by the simulation software. In essence, the module shown in Step 3 is a repository for the information read in from the interface. The files and data in this module have a fixed structure that is required by the simulation software.

Another routine is executed to build simulation objects from the templates, step 4. The code for implementing this module, in an embodiment of the present invention, is shown and described below. During this step, data, files, and information are read into the simulation software from the module of step 3 to build a simulation. Also, during this step, the simulation software is launched.

Next, a simulation is run, step 5. At this point the simulation model and all associated files exist. The simulation is run and displayed on a computer screen.

A routine then exports values from the simulation routine to a Reports spreadsheet, such as for example, an MS Excel® spreadsheet, step 6. The Reports spreadsheet may contain a user input interface to allow for the generation of custom reports based on customer-specific data. In some embodiments, the reports may be generated in the simulation software in addition to or in place of a report spreadsheet.

As is shown in Figure 2, an input interface spreadsheet is configured to accept certain set-up data, such as the name of the project being modeled, file management information, time of the simulation run, etc. The user may, for example, enter the directory where the simulation software is saved. The user may also input a project name, and the user may input a simulation time in a convenient unit, such as days.

The input spreadsheet can be configured with a soft button or other selection mechanism, such as the "Create Project" button shown in Figure 2. When the user selects the "Create Project" button the appropriate subdirectories appear and the required files are copied. Figure 3 illustrates files and directories that are created in the background when the "Create Project" button is selected in one embodiment. For example, a CBM directory may be created and appropriate templates copied. In addition, a project directory may be created for a specific project and populated with appropriate templates and files, such as the CBM.XLS, CMB-Transfer.XLS, and the CBM_Reports.XLS, as shown.

As is shown in Figure 2, various other soft buttons may be employed to select various user interfaces and user requested processes. The "Shift Data" button opens a user interface as depicted in Figures 4-4C. This interface also takes the form of a spreadsheet. The user populates this spreadsheet with appropriate shift patterns, an example of which is shown in Figures 4-4C.

After the appropriate Shift data is entered, the user may select the "Line Configuration" button, shown in Figure 2. Upon selecting this button, the screen in Figures 5-5C appears. The user may then select the equipment to be added to the line. The user also can select where to insert a piece of equipment. As is shown in Figure 6, a drop down menu can be employed to facilitate populating this spreadsheet.

Adding a piece of equipment results in the addition of a row in the line configuration spreadsheet and adds to the number of lines which are stored in process cells of the board spreadsheet, discussed below. (See Figs. 7-7F). The process of adding equipment is continued until a line is completed. By using an insert procedure with the input of a row number and a delete procedure, which works accordingly, the first version of the line can be manipulated. Clicking on "Main" returns to the main user interface.

As the user enters a specific piece of equipment at a particular cell of the column (or row), other cells in the same column (or row) are automatically populated with values that correspond to specific operating parameters for that piece of equipment. These parameters may include, but are not limited to, Machine Cycle Time, Mean Time Between Manual Assists, Mean Time to Assist, Shift Pattern (i.e. how many shifts per day the machine will be operated for), Mean Time Between Failure, etc. Figures 5-5C illustrates, without limitation, some of the parameters that may be associated with a specific piece of equipment.

The values for these parameters may be stored in an object parameter database, which may take the form of another spreadsheet, that can be periodically updated based on real-world experience. One advantage of maintaining a parameter database as a separate spreadsheet is that it may be updated over time with empirically derived, real-world data, that is gathered in the field for each component and each line configuration that is developed. For example, the components of an assembly line can be configured to automatically track the data required for these parameters. That data can then be used to update the second spreadsheet. The tool may

use the latest data available at the conclusion of each project, where the tool is employed, to update the object database parameters.

In addition to inputting a specific line configuration through the "Line Configuration" spreadsheet interface and shift data through the "Shift Data" interface, the user can input specific board data by selecting the "Board Data" button at the main user interface shown in Figure 2. As is shown in Figure 8 the user may enter cycle times for every piece of equipment, enter board specific data, enter a board to use for the simulation, and then return to the main menu by clicking on "Main."

After all the input spreadsheet interfaces are populated with the necessary data, the user executes one or more routines that build simulation objects from the data in the spreadsheet. As is shown in Figure 2, the spreadsheet can be configured with a soft button, such as the "Create Simulation" button, to build the simulation objects and, in some embodiments, may automatically run a simulation; in other embodiments, the user may be prompted to run the simulation after the simulation objects are built.

Figure 9 illustrates a screen view that is reached by selecting the "Create Simulation" button. At this point, all data out of the simulation is transferred from the input interfaces to the CBM_Transfer.XLS module, a Base.Mod is copied to the project directory, and Witness® 2000 or other simulation software is opened. The simulation is started and the line or other electronics assembly configuration is built on the screen.

The process for building the simulation objects may be streamlined by creating templates in the simulation software for the simulation objects. Set forth below is an example of a template for a simulation object for a cluster of 3 Siemens HS-50 machines that may be used with Witness ® 2000.

HS-50 Template

* * * Begin Template * * *

SELECT

HS50Cluster310101

09873021.060101

NAME OF MODULE: HS50Cluster310101;

TITLE: ;

INPUT ELEMENT: Undefined;

OUTPUT ELEMENT: Undefined;

FILENAME: C:\IanRawles\Projects\CBMMonday\H50Cluster3_L1.mdl;

WALK: No;

LINK_TO_FILE: No;

USE CYCLE: No;

CYCLE TIME: Undefined;

ELEMENT PLACEMENT: -1;

REPORTING: Individual;

BEGIN_MODULE_DETAIL

HS50

NAME OF MODULE: HS50;

TITLE: ;

INPUT ELEMENT: Undefined;

OUTPUT ELEMENT: Undefined;

ACTIONS, Initialise

Add

!1=asynch, 2=synch

HS50.SWITCHS25AS = 1

HS50.SWITCHS25SINGDUAL = 2

!

!Put flags to one if maintenance stops line, 0 if maintenance is smarter and performed by operator as

! conditions allow

! 1= daily, 2-weekly, 3=long term.

!

HS50.Maintainflag (1) = 1

HS50.Maintainflag (2) = 1

HS50.Maintainflag (3) = 1

!

! Configuration for single vs. dual conveyor

!

!SET QUANTITY OF Conveyor001 TO SWITCHS25SINGDUAL

!SET QUANTITY OF Conveyor002 TO SWITCHS25SINGDUAL

!SET QUANTITY OF Go TO SWITCHS25SINGDUAL

!SET QUANTITY OF Cycletimes TO SWITCHS25SINGDUAL

!SET QUANTITY OF Setuptimes TO SWITCHS25SINGDUAL

!SET QUANTITY OF Machine001 TO SWITCHS25SINGDUAL

End Actions

WALK: No;

LINK_TO_FILE: No;

USE CYCLE: No;

CYCLE TIME: Undefined;

ELEMENT PLACEMENT: -1;

REPORTING: Individual;

BEGIN_MODULE_DETAIL

Conveyor001

NAME OF CONVEYOR: Conveyor001;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

INPUT RULE: Wait;

OUTPUT RULE: Wait;

CYCLE TIME: 2.0;

BREAKDOWNS: No;

PRIORITY: Lowest;

LABOR:

Repair: None;

09873021-060101

2000P07660US01

END

REPORTING: Individual;

SHIFT: Undefined;

RESTART DELAY: Undefined;

END Conveyor001

Machine001

NAME OF MACHINE: Machine001;

QUANTITY: 1;

TYPE: Single;

PRIORITY: Lowest;

LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: 3.0 !Setuptimes (N);

INPUT RULE: PULL from Conveyor001(N) at Front;

OUTPUT RULE: IF Go (N) = 1

PUSH to Conveyor002(N) at Rear

ELSE

Wait

ENDIF;

Output_From: Front;

REPORTING: Individual;

SHIFT: Undefined,0,0;

09873021.060101

END Machine001

Conveyor002

NAME OF CONVEYOR: Conveyor002;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

ACTIONS, Join

Add

Go (N) = 0

End Actions

INPUT RULE: Wait;

OUTPUT RULE: Wait;

CYCLE TIME: 2.0;

BREAKDOWNS: No;

PRIORITY: Lowest;

LABOR:

Repair: None;

END

REPORTING: Individual;

SHIFT: Undefined;

RESTART DELAY: Undefined;

END Conveyor002

Head

NAME OF MACHINE: Head;

QUANTITY: 1;

09873021.060101

TYPE: General(Multi-Cycle);

PRIORITY: Lowest;

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE_DETAIL

Cycle number: 1

* Cycle time: 1.0;

* Input quantity: 1;

* Finish quantity: 1;

* Description: Cycle number 1

ACTIONS, Finish

Add

IF TYPE = Dummy1

Go (1) = 1

ELSEIF TYPE = Dummy2

Go (2) = 1

ELSE

STOP

ENDIF

IF TYPE = Dummy1

Headwhere1 = "Top Lane"

ELSE

Headwhere1 = "Bottom Lane"

ENDIF

End Actions

LABOR:

Cycle: None;

Pre-empt level: None;

09873021.060101

END

Input rule: IF ISTATE (Machine001(1)) = 3 AND Go (1) = 0

PULL from Dummy1 out of WORLD

ELSE

Wait

ENDIF;

Output Rule: Wait;

Output_From: Front;

Output_Quantity: 0;

Cycle number: 2

* Cycle time: 0.01;

* Input quantity: 0;

* Finish quantity: 1;

* Description: Cycle2

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: Wait;

Output Rule: IF Go (1) = 0

PUSH to SHIP

ELSE

Wait

ENDIF;

Output_From: Front;

END CYCLE_DETAIL

BREAKDOWN_DETAIL

Breakdown number: 1

* Mode: Busy time;

* Down interval: Undefined;

* Repair time: Undefined;

* Scrap part: No;

09873021-060101

* Description: MTBF
 * Setup on repair: No;
 * Check at cycle start only: Yes;
 DIALOG DISPLAY NUMBER: 0;
 ACTIONS, Breakdown
 Add

BREAKDWN (Conveyor001(1))
 BREAKDWN (Machine001(1))
 BREAKDWN (Conveyor002(1))
 BREAKDWN (Machine002(1))
 BREAKDWN (Head2)

End Actions
 ACTIONS, Repair
 Add

REPAIR (Conveyor001(1))
 REPAIR (Machine001(1))
 REPAIR (Conveyor002(1))
 REPAIR (Machine002(1))
 REPAIR (Head2)

End Actions
 LABOR:

Repair: None;
 END
 Breakdown number: 2
 * Mode: Busy time;
 * Down interval: NORMAL (3600.0,500,48);
 * Repair time: 3.5 * 60;
 * Scrap part: No;
 * Description: MTBA

09873021.060101

* Setup on repair: No;
* Check at cycle start only: Yes;
DIALOG DISPLAY NUMBER: 1;

ACTIONS, Breakdown

Add

BREAKDOWN (Machine001(1))

End Actions

ACTIONS, Repair

Add

REPAIR (Machine001(1))

End Actions

LABOR:

Repair: None;

END

Breakdown number: 3

* Mode: Busy time;

* Down interval: TRIANGLE (2.5 * 1440 * 60,3 * 1440 * 60,3.5 * 1440 *
60,72);

* Repair time: 120.0;

* Scrap part: No;

* Description: Nozzle ID Error

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 2;

LABOR:

Repair: None;

END

END BREAKDOWN_DETAIL

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Head

Go

NAME OF VARIABLE: Go;

QUANTITY: 4;

REPORTING: Yes;

END Go

SWITCHS25AS

NAME OF VARIABLE: SWITCHS25AS;

QUANTITY: 1;

REPORTING: Yes;

END SWITCHS25AS

Cycletimes

NAME OF VARIABLE: Cycletimes;

QUANTITY: 1;

REPORTING: Yes;

END Cycletimes

Setuptimes

NAME OF VARIABLE: Setuptimes;

QUANTITY: 1;

REPORTING: Yes;

END Setuptimes

09873021.060101

SWITCHS25SINGDUAL

NAME OF VARIABLE: SWITCHS25SINGDUAL;

QUANTITY: 1;

REPORTING: Yes;

END SWITCHS25SINGDUAL

Operator

NAME OF LABOR: Operator;

QUANTITY no shift: 1;

REPORTING: Group;

END Operator

Maintainflag

NAME OF VARIABLE: Maintainflag;

QUANTITY: 3;

REPORTING: Yes;

END Maintainflag

Day

NAME OF SHIFT: Day;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

09873021-060101

ACTIONS, Start

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (1) = 1

BREAKDWN (Head)

ENDIF

End Actions

SHIFT DATA: 28200.000000,600.000000,0.000000;

REPORTING: Yes;

END Day

Weekly

NAME OF SHIFT: Weekly;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (2) = 1

BREAKDWN (Head)

ENDIF

End Actions

09873021.060101

2000P07660US01

SHIFT DATA: 6002880.000000,1920.000000,0.000000;

REPORTING: Yes;

END Weekly

LongtermFactor

NAME OF SHIFT: LongtermFactor;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (3) = 1

BREAKDOWN (Head)

ENDIF

End Actions

SHIFT DATA: 596575.000000,8225.280273,0.000000;

REPORTING: Yes;

END LongtermFactor

Machine002

NAME OF MACHINE: Machine002;

QUANTITY: 1;

TYPE: Single;

09873021.060101

PRIORITY: Lowest;

LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: 3.0 !Setuptimes (N);

INPUT RULE: PULL from Conveyor002(N) at Front;

OUTPUT RULE: IF Go (N + 2) = 1

PUSH to Conveyor003(N) at Rear

ELSE

Wait

ENDIF;

Output_From: Front;

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Machine002

Head2

NAME OF MACHINE: Head2;

QUANTITY: 1;

TYPE: General(Multi-Cycle);

PRIORITY: Lowest;

DISCRETE LINKS :

Fill: None

END

09873021.060101

DISCRETE LINKS :

Empty: None

END

CYCLE_DETAIL

Cycle number: 1

* Cycle time: 1.0;

* Input quantity: 1;

* Finish quantity: 1;

* Description: Cycle number 1

ACTIONS, Finish

Add

IF TYPE = Dummy3

Go (3) = 1

ELSEIF TYPE = Dummy4

Go (4) = 1

ELSE

STOP

ENDIF

IF TYPE = Dummy3

Headwhere2 = "Top Lane"

ELSEIF TYPE = Dummy4

Headwhere2 = "Bottom Lane"

ENDIF

End Actions

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: IF ISTATE (Machine002(1)) = 3 AND Go (3) = 0

PULL from Dummy3 out of WORLD

ELSE

Wait

09873021.060101

ENDIF;

Output Rule: PUSH to SHIP;

Output_From: Front;

Output_Quantity: 0;

Cycle number: 2

* Cycle time: 0.01;

* Input quantity: 0;

* Finish quantity: 1;

* Description: Cycle2

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: Wait;

Output Rule: IF Go (3) = 0 OR Go (4) = 0

PUSH to SHIP

ELSE

Wait

ENDIF;

Output_From: Front;

END CYCLE_DETAIL

BREAKDOWN_DETAIL

Breakdown number: 1

* Mode: Busy time;

* Down interval: Undefined;

* Repair time: Undefined;

* Scrap part: No;

* Description: MTBF

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 0;

ACTIONS, Breakdown

09873021.060101

Add

BREAKDWN (Conveyor001(1))
 BREAKDWN (Machine001(1))
 BREAKDWN (Conveyor002(1))
 BREAKDWN (Machine002(1))
 BREAKDWN (Head)

End Actions

ACTIONS, Repair

Add

REPAIR (Conveyor001(1))
 REPAIR (Machine001(1))
 REPAIR (Conveyor002(1))
 REPAIR (Machine002(1))
 REPAIR (Head)

End Actions

LABOR:

Repair: None;

END

Breakdown number: 2

* Mode: Busy time;

* Down interval: NORMAL (3600.0,500,48);

* Repair time: 3.5 * 60;

* Scrap part: No;

* Description: MTBA

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 1;

ACTIONS, Breakdown

Add

BREAKDWN (Machine002(1))

End Actions

ACTIONS, Repair

09873021.060101

Add

REPAIR (Machine002(1))

End Actions

LABOR:

Repair: None;

END

Breakdown number: 3

* Mode: Busy time;

* Down interval: TRIANGLE (2.5 * 1440 * 60,3 * 1440 * 60,3.5 * 1440 *

60,72);

* Repair time: 120.0;

* Scrap part: No;

* Description: Nozzle ID Error

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 2;

LABOR:

Repair: None;

END

END BREAKDOWN_DETAIL

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Head2

Conveyor003

NAME OF CONVEYOR: Conveyor003;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

09873021.060101

INPUT RULE: Wait;
OUTPUT RULE: Wait;
CYCLE TIME: 2.0;
BREAKDOWNS: No;
PRIORITY: Lowest;
LABOR:
Repair: None;
END
REPORTING: Individual;
SHIFT: Undefined;
RESTART DELAY: Undefined;

END Conveyor003

Dummy1

NAME OF PART: Dummy1;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy1

Dummy2

NAME OF PART: Dummy2;

09873021.060101

2000P07660US01

TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy2

Dummy3

NAME OF PART: Dummy3;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy3

Dummy4

NAME OF PART: Dummy4;
TYPE: Variable attributes;
GROUP NUMBER: 1;

09873021.060101

2000P07660US01

MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy4

Headwhere1

NAME OF VARIABLE: Headwhere1;
QUANTITY: 1;
REPORTING: Yes;

END Headwhere1

Headwhere2

NAME OF VARIABLE: Headwhere2;
QUANTITY: 1;
REPORTING: Yes;

END Headwhere2

Pie001

NAME OF PIE_CHART: Pie001;
QUANTITY: 1;
STATE_VALUES:
* Name: Head;

09873021.060101

2000P07660US01

* Type: standard;
* Report by: on_shift_time;
* Values: group;
REFRESH INTERVAL: 3600;
REPORTING: Individual;
PULL_OUT_SEGMENT: 0;
SHIFT: Undefined;

END Pie001

Pie002

NAME OF PIE_CHART: Pie002;
QUANTITY: 1;
STATE_VALUES:
* Name: Head2;
* Type: standard;
* Report by: on_shift_time;
* Values: group;
REFRESH INTERVAL: 3600;
REPORTING: Individual;
PULL_OUT_SEGMENT: 0;
SHIFT: Undefined;

END Pie002

END_MODULE_DETAIL

END HS50

HS51

09873021.060101

NAME OF MODULE: HS51;

TITLE: ;

INPUT ELEMENT: Undefined;

OUTPUT ELEMENT: Undefined;

ACTIONS, Initialise

Add

!1=asynch, 2=synch

HS51.SWITCHS25AS = 1

HS51.SWITCHS25SINGDUAL = 2

!

!Put flags to one if maintenance stops line, 0 if maintenance is smarter and performed by operator as

! conditions allow

! 1= daily, 2-weekly, 3=long term.

!

HS51.Maintainflag (1) = 1

HS51.Maintainflag (2) = 1

HS51.Maintainflag (3) = 1

!

! Configuration for single vs. dual conveyor

!

!SET QUANTITY OF Conveyor001 TO SWITCHS25SINGDUAL

!SET QUANTITY OF Conveyor002 TO SWITCHS25SINGDUAL

!SET QUANTITY OF Go TO SWITCHS25SINGDUAL

!SET QUANTITY OF Cyclicetimes TO SWITCHS25SINGDUAL

!SET QUANTITY OF Setuptimes TO SWITCHS25SINGDUAL

!SET QUANTITY OF Machine001 TO SWITCHS25SINGDUAL

End Actions

WALK: No;

LINK_TO_FILE: No;

USE CYCLE: No;

CYCLE TIME: Undefined;

09873021.050101

ELEMENT PLACEMENT: -1;

REPORTING: Individual;

BEGIN_MODULE_DETAIL

Conveyor001

NAME OF CONVEYOR: Conveyor001;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

INPUT RULE: Wait;

OUTPUT RULE: Wait;

CYCLE TIME: 2.0;

BREAKDOWNS: No;

PRIORITY: Lowest;

LABOR:

Repair: None;

END

REPORTING: Individual;

SHIFT: Undefined;

RESTART DELAY: Undefined;

END Conveyor001

Machine001

NAME OF MACHINE: Machine001;

QUANTITY: 1;

TYPE: Single;

PRIORITY: Lowest;

LABOR:

Cycle: None;

09873021.060101

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: 3.0 !Setuptimes (N);

INPUT RULE: PULL from Conveyor001(N) at Front;

OUTPUT RULE: IF Go (N) = 1

PUSH to Conveyor002(N) at Rear

ELSE

Wait

ENDIF;

Output_From: Front;

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Machine001

Conveyor002

NAME OF CONVEYOR: Conveyor002;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

ACTIONS, Join

Add

Go (N) = 0

End Actions

INPUT RULE: Wait;

09873021.060101

OUTPUT RULE: Wait;
CYCLE TIME: 2.0;
BREAKDOWNS: No;
PRIORITY: Lowest;
LABOR:
 Repair: None;
END
REPORTING: Individual;
SHIFT: Undefined;
RESTART DELAY: Undefined;

END Conveyor002

Head

NAME OF MACHINE: Head;
QUANTITY: 1;
TYPE: General(Multi-Cycle);
 PRIORITY: Lowest;
DISCRETE LINKS :
 Fill: None

END

DISCRETE LINKS :
 Empty: None

END

CYCLE_DETAIL

Cycle number: 1
* Cycle time: 1.0;
* Input quantity: 1;
* Finish quantity: 1;
* Description: Cycle number 1

ACTIONS, Finish

Add

```

IF TYPE = Dummy1
    Go (1) = 1
ELSEIF TYPE = Dummy2
    Go (2) = 1
ELSE
    STOP
ENDIF
IF TYPE = Dummy1
    Headwhere1 = "Top Lane"
ELSE
    Headwhere1 = "Bottom Lane"
ENDIF

```

End Actions

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: IF ISTATE (Machine001(1)) = 3 AND Go (1) = 0

PULL from Dummy1 out of WORLD

ELSE

Wait

ENDIF;

Output Rule: Wait;

Output_From: Front;

Output_Quantity: 0;

Cycle number: 2

* Cycle time: 0.01;

* Input quantity: 0;

* Finish quantity: 1;

* Description: Cycle2

LABOR:

09873021.060101

Cycle: None;
 Pre-empt level: None;
 END
 Input rule: Wait;
 Output Rule: IF Go (1) = 0
 PUSH to SHIP
 ELSE
 Wait
 ENDIF;

Output_From: Front;
 END CYCLE_DETAIL
 BREAKDOWN_DETAIL

Breakdown number: 1
 * Mode: Busy time;
 * Down interval: Undefined;
 * Repair time: Undefined;
 * Scrap part: No;
 * Description: MTBF
 * Setup on repair: No;
 * Check at cycle start only: Yes;
 DIALOG DISPLAY NUMBER: 0;
 ACTIONS, Breakdown
 Add

BREAKDWN (Conveyor001(1))
 BREAKDWN (Machine001(1))
 BREAKDWN (Conveyor002(1))
 BREAKDWN (Machine002(1))
 BREAKDWN (Head2)

End Actions

09873021.060101

ACTIONS, Repair

Add

REPAIR (Conveyor001(1))

REPAIR (Machine001(1))

REPAIR (Conveyor002(1))

REPAIR (Machine002(1))

REPAIR (Head2)

End Actions

LABOR:

Repair: None;

END

Breakdown number: 2

* Mode: Busy time;

* Down interval: NORMAL (3600.0,500,48);

* Repair time: 3.5 * 60;

* Scrap part: No;

* Description: MTBA

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 1;

ACTIONS, Breakdown

Add

BREAKDWN (Machine001(1))

End Actions

ACTIONS, Repair

Add

REPAIR (Machine001(1))

End Actions

LABOR:

Repair: None;

END

Breakdown number: 3

09873021.060101

* Mode: Busy time;

* Down interval: TRIANGLE (2.5 * 1440 * 60,3 * 1440 * 60,3.5 * 1440 * 60,72);

* Repair time: 120.0;

* Scrap part: No;

* Description: Nozzle ID Error

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 2;

LABOR:

Repair: None;

END

END BREAKDOWN_DETAIL

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Head

Go

NAME OF VARIABLE: Go;

QUANTITY: 4;

REPORTING: Yes;

END Go

SWITCHS25AS

NAME OF VARIABLE: SWITCHS25AS;

QUANTITY: 1;

REPORTING: Yes;

09873021.060101

END SWITCHS25AS

Cycletimes

NAME OF VARIABLE: Cycletimes;

QUANTITY: 1;

REPORTING: Yes;

END Cycletimes

Setuptimes

NAME OF VARIABLE: Setuptimes;

QUANTITY: 1;

REPORTING: Yes;

END Setuptimes

SWITCHS25SINGDUAL

NAME OF VARIABLE: SWITCHS25SINGDUAL;

QUANTITY: 1;

REPORTING: Yes;

END SWITCHS25SINGDUAL

Operator

NAME OF LABOR: Operator;

QUANTITY no shift: 1;

REPORTING: Group;

09873021.060101

END Operator

Maintainflag

NAME OF VARIABLE: Maintainflag;

QUANTITY: 3;

REPORTING: Yes;

END Maintainflag

Day

NAME OF SHIFT: Day;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (1) = 1

BREAKDWN (Head)

ENDIF

End Actions

SHIFT DATA: 28200.000000,600.000000,0.000000;

REPORTING: Yes;

END Day

09873021.060101

Weekly

NAME OF SHIFT: Weekly;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (2) = 1

BREAKDWN (Head)

ENDIF

End Actions

SHIFT DATA: 6002880.000000,1920.000000,0.000000;

REPORTING: Yes;

END Weekly

LongtermFactor

NAME OF SHIFT: LongtermFactor;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

Add

REPAIR (Head)

09873021.060101

End Actions

ACTIONS, End

Add

IF Maintainflag (3) = 1

BREAKDOWN (Head)

ENDIF

End Actions

SHIFT DATA: 596575.000000,8225.280273,0.000000;

REPORTING: Yes;

END LongtermFactor

Machine002

NAME OF MACHINE: Machine002;

QUANTITY: 1;

TYPE: Single;

PRIORITY: Lowest;

LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: 3.0 !Setuptimes (N);

INPUT RULE: PULL from Conveyor002(N) at Front;

OUTPUT RULE: IF Go (N + 2) = 1

PUSH to Conveyor003(N) at Rear

ELSE

09873021-060101

Wait

ENDIF;

Output_From: Front;

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Machine002

Head2

NAME OF MACHINE: Head2;

QUANTITY: 1;

TYPE: General(Multi-Cycle);

PRIORITY: Lowest;

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE_DETAIL

Cycle number: 1

* Cycle time: 1.0;

* Input quantity: 1;

* Finish quantity: 1;

* Description: Cycle number 1

ACTIONS, Finish

Add

IF TYPE = Dummy3

Go (3) = 1

ELSEIF TYPE = Dummy4

Go (4) = 1

09873001.060101


```

ELSE
    STOP
ENDIF
IF TYPE = Dummy3
    Headwhere2 = "Top Lane"
ELSEIF TYPE = Dummy4
    Headwhere2 = "Bottom Lane"
ENDIF

```

End Actions

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: IF ISTATE (Machine002(1)) = 3 AND Go (3) = 0

PULL from Dummy3 out of WORLD

ELSE

Wait

ENDIF;

Output Rule: PUSH to SHIP;

Output_From: Front;

Output_Quantity: 0;

Cycle number: 2

* Cycle time: 0.01;

* Input quantity: 0;

* Finish quantity: 1;

* Description: Cycle2

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: Wait;

Output Rule: IF Go (3) = 0

00873021.060101

PUSH to SHIP

ELSE

Wait

ENDIF;

Output_From: Front;

END CYCLE_DETAIL

BREAKDOWN_DETAIL

Breakdown number: 1

* Mode: Busy time;

* Down interval: Undefined;

* Repair time: Undefined;

* Scrap part: No;

* Description: MTBF

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 0;

ACTIONS, Breakdown

Add

BREAKDWN (Conveyor001(1))

BREAKDWN (Machine001(1))

BREAKDWN (Conveyor002(1))

BREAKDWN (Machine002(1))

BREAKDWN (Head)

End Actions

ACTIONS, Repair

Add

REPAIR (Conveyor001(1))

REPAIR (Machine001(1))

REPAIR (Conveyor002(1))

REPAIR (Machine002(1))

REPAIR (Head)

End Actions

09873021.050101

LABOR:

Repair: None;

END

Breakdown number: 2

* Mode: Busy time;

* Down interval: NORMAL (3600.0,500,48);

* Repair time: 3.5 * 60;

* Scrap part: No;

* Description: MTBA

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 1;

ACTIONS, Breakdown

Add

BREAKDOWN (Machine002(1))

End Actions

ACTIONS, Repair

Add

REPAIR (Machine002(1))

End Actions

LABOR:

Repair: None;

END

Breakdown number: 3

* Mode: Busy time;

* Down interval: TRIANGLE (2.5 * 1440 * 60,3 * 1440 * 60,3.5 * 1440 * 60,72);

* Repair time: 120.0;

* Scrap part: No;

* Description: Nozzle ID Error

* Setup on repair: No;

* Check at cycle start only: Yes;

09873021.060101

2000P07660US01

DIALOG DISPLAY NUMBER: 2;

LABOR:

Repair: None;

END

END BREAKDOWN_DETAIL

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Head2

Conveyor003

NAME OF CONVEYOR: Conveyor003;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

INPUT RULE: Wait;

OUTPUT RULE: Wait;

CYCLE TIME: 2.0;

BREAKDOWNS: No;

PRIORITY: Lowest;

LABOR:

Repair: None;

END

REPORTING: Individual;

SHIFT: Undefined;

RESTART DELAY: Undefined;

END Conveyor003

Dummy1

09873021-060101

NAME OF PART: Dummy1;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy1

Dummy2

NAME OF PART: Dummy2;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy2

Dummy3

NAME OF PART: Dummy3;

09873021.060101

TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy3

Dummy4

NAME OF PART: Dummy4;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy4

Headwhere1

NAME OF VARIABLE: Headwhere1;
QUANTITY: 1;
REPORTING: Yes;

09873021.060101

END Headwhere1

Headwhere2

NAME OF VARIABLE: Headwhere2;

QUANTITY: 1;

REPORTING: Yes;

END Headwhere2

Pie001

NAME OF PIE_CHART: Pie001;

QUANTITY: 1;

STATE_VALUES:

* Name: Head;

* Type: standard;

* Report by: on_shift_time;

* Values: group;

REFRESH INTERVAL: 3600;

REPORTING: Individual;

PULL_OUT_SEGMENT: 0;

SHIFT: Undefined;

END Pie001

Pie002

NAME OF PIE_CHART: Pie002;

QUANTITY: 1;

STATE_VALUES:

09873021.060101

```

* Name: Head2;
* Type: standard;
* Report by: on_shift_time;
* Values: group;
REFRESH INTERVAL: 3600;
REPORTING: Individual;
PULL_OUT_SEGMENT: 0;
SHIFT: Undefined;

```

END Pie002

END_MODULE_DETAIL

END HS51

HS52

```

NAME OF MODULE: HS52;
TITLE: ;
INPUT ELEMENT: Undefined;
OUTPUT ELEMENT: Undefined;
ACTIONS, Initialise
Add

```

!1=asynch, 2=synch

HS52.SWITCHS25AS = 1

HS52.SWITCHS25SINGDUAL = 2

!

!Put flags to one if maintenance stops line, 0 if maintenance is smarter and performed by operator as

! conditions allow

! 1= daily, 2-weekly, 3=long term.

!

HS52.Maintainflag (1) = 1

HS52.Maintainflag (2) = 1

HS52.Maintainflag (3) = 1

!

! Configuration for single vs. dual conveyor

!

!SET QUANTITY OF Conveyor001 TO SWITCHS25SINGDUAL

!SET QUANTITY OF Conveyor002 TO SWITCHS25SINGDUAL

!SET QUANTITY OF Go TO SWITCHS25SINGDUAL

!SET QUANTITY OF Cycletimes TO SWITCHS25SINGDUAL

!SET QUANTITY OF Setuptimes TO SWITCHS25SINGDUAL

!SET QUANTITY OF Machine001 TO SWITCHS25SINGDUAL

End Actions

WALK: No;

LINK_TO_FILE: No;

USE CYCLE: No;

CYCLE TIME: Undefined;

ELEMENT PLACEMENT: -1;

REPORTING: Individual;

BEGIN_MODULE_DETAIL

Conveyor001

NAME OF CONVEYOR: Conveyor001;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

INPUT RULE: Wait;

OUTPUT RULE: Wait;

CYCLE TIME: 2.0;

BREAKDOWNS: No;

PRIORITY: Lowest;

09873021-060101

LABOR:

Repair: None;

END

REPORTING: Individual;

SHIFT: Undefined;

RESTART DELAY: Undefined;

END Conveyor001

Machine001

NAME OF MACHINE: Machine001;

QUANTITY: 1;

TYPE: Single;

PRIORITY: Lowest;

LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: 3.0 !Setuptimes (N);

INPUT RULE: PULL from Conveyor001(N) at Front;

OUTPUT RULE: IF Go (N) = 1

PUSH to Conveyor002(N) at Rear

ELSE

Wait

ENDIF;

Output_From: Front;

09873021-060101

REPORTING: Individual;
SHIFT: Undefined,0,0;

END Machine001

Conveyor002

NAME OF CONVEYOR: Conveyor002;
QUANTITY: 1;
TYPE: Queuing;
PART LENGTH: 1;
MAX CAPACITY: 1;
ACTIONS, Join
Add

Go (N) = 0

End Actions

INPUT RULE: !IF ISTATE (Machine001(1)) = 3

!PULL from Dummy1 out of WORLD

!ELSEIF ISTATE (Machine001(2)) = 3

!PULL from Dummy2 out of WORLD

!ELSE

!Wait

!ENDIF

Wait;

OUTPUT RULE: Wait;

CYCLE TIME: 2.0;

BREAKDOWNS: No;

PRIORITY: Lowest;

LABOR:

Repair: None;

END

REPORTING: Individual;

SHIFT: Undefined;
RESTART DELAY: Undefined;

END Conveyor002

Head

NAME OF MACHINE: Head;
QUANTITY: 1;
TYPE: General(Multi-Cycle);
PRIORITY: Lowest;

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE_DETAIL

Cycle number: 1

* Cycle time: 1.0;

* Input quantity: 1;

* Finish quantity: 1;

* Description: Cycle number 1

ACTIONS, Finish

Add

IF TYPE = Dummy1

Go (1) = 1

ELSEIF TYPE = Dummy2

Go (2) = 1

ELSE

STOP

ENDIF

IF TYPE = Dummy1

Headwhere1 = "Top Lane"

ELSE

Headwhere1 = "Bottom Lane"

ENDIF

End Actions

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: IF ISTATE (Machine001(1)) = 3 AND Go (1) = 0

PULL from Dummy1 out of WORLD

ELSE

Wait

ENDIF;

Output Rule: Wait;

Output_From: Front;

Output_Quantity: 0;

Cycle number: 2

* Cycle time: 0.01;

* Input quantity: 0;

* Finish quantity: 1;

* Description: Cycle2

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: Wait;

Output Rule: IF Go (1) = 0

PUSH to SHIP

ELSE

Wait

09873021.060101

ENDIF;

Output_From: Front;

END CYCLE_DETAIL

BREAKDOWN_DETAIL

Breakdown number: 1

* Mode: Busy time;

* Down interval: Undefined;

* Repair time: Undefined;

* Scrap part: No;

* Description: MTBF

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 0;

ACTIONS, Breakdown

Add

!

BREAKDWN (Conveyor001(1))

BREAKDWN (Machine001(1))

BREAKDWN (Conveyor002(1))

BREAKDWN (Machine002(1))

BREAKDWN (Head2)

!

!

End Actions

ACTIONS, Repair

Add

REPAIR (Conveyor001(1))

REPAIR (Machine001(1))

REPAIR (Conveyor002(1))

REPAIR (Machine002(1))

REPAIR (Head2)

End Actions

09873021.060101

LABOR:

Repair: None;

END

Breakdown number: 2

* Mode: Busy time;

* Down interval: NORMAL (3600.0,500,48);

* Repair time: 3.5 * 60;

* Scrap part: No;

* Description: MTBA

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 1;

ACTIONS, Breakdown

Add

BREAKDWN (Machine001(1))

End Actions

ACTIONS, Repair

Add

REPAIR (Machine001(1))

End Actions

LABOR:

Repair: None;

END

Breakdown number: 3

* Mode: Busy time;

* Down interval: TRIANGLE (2.5 * 1440 * 60,3 * 1440 * 60,3.5 * 1440 * 60,72);

* Repair time: 120.0;

* Scrap part: No;

* Description: Nozzle ID Error

* Setup on repair: No;

* Check at cycle start only: Yes;

09873021.060101

DIALOG DISPLAY NUMBER: 2;

LABOR:

Repair: None;

END

END BREAKDOWN_DETAIL

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Head

Go

NAME OF VARIABLE: Go;

QUANTITY: 4;

REPORTING: Yes;

END Go

SWITCHS25AS

NAME OF VARIABLE: SWITCHS25AS;

QUANTITY: 1;

REPORTING: Yes;

END SWITCHS25AS

Cycletimes

NAME OF VARIABLE: Cycletimes;

QUANTITY: 1;

REPORTING: Yes;

09873021.060101

END Cycletimes

Setuptimes

NAME OF VARIABLE: Setuptimes;

QUANTITY: 1;

REPORTING: Yes;

END Setuptimes

SWITCHS25SINGDUAL

NAME OF VARIABLE: SWITCHS25SINGDUAL;

QUANTITY: 1;

REPORTING: Yes;

END SWITCHS25SINGDUAL

Operator

NAME OF LABOR: Operator;

QUANTITY no shift: 1;

REPORTING: Group;

END Operator

Maintainflag

NAME OF VARIABLE: Maintainflag;

QUANTITY: 3;

REPORTING: Yes;

09873021.060101

END Maintainflag

Day

NAME OF SHIFT: Day;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (1) = 1

BREAKDWN (Head)

ENDIF

End Actions

SHIFT DATA: 28200.000000,600.000000,0.000000;

REPORTING: Yes;

END Day

Weekly

NAME OF SHIFT: Weekly;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

09873021.060101

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (2) = 1

BREAKDWN (Head)

ENDIF

End Actions

SHIFT DATA: 6002880.000000,1920.000000,0.000000;

REPORTING: Yes;

END Weekly

LongtermFactor

NAME OF SHIFT: LongtermFactor;

TYPE OF SHIFT: main;

OFFSET:

Working time: 0.000000;

Rest time: 0.000000;

ACTIONS, Start

Add

REPAIR (Head)

End Actions

ACTIONS, End

Add

IF Maintainflag (3) = 1

BREAKDWN (Head)

ENDIF

End Actions

SHIFT DATA: 596575.000000,8225.280273,0.000000;

09873021.060101

REPORTING: Yes;

END LongtermFactor

Machine002

NAME OF MACHINE: Machine002;

QUANTITY: 1;

TYPE: Single;

PRIORITY: Lowest;

LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: 3.0 !Setuptimes (N);

INPUT RULE: PULL from Conveyor002(N) at Front;

OUTPUT RULE: IF Go (N + 2) = 1

PUSH to Conveyor003(N) at Rear

ELSE

Wait

ENDIF;

Output_From: Front;

REPORTING: Individual;

SHIFT: Undefined,0,0;

END Machine002

09873021.060101

Head2

NAME OF MACHINE: Head2;

QUANTITY: 1;

TYPE: General(Multi-Cycle);

PRIORITY: Lowest;

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE_DETAIL

Cycle number: 1

* Cycle time: 1.0;

* Input quantity: 1;

* Finish quantity: 1;

* Description: Cycle number 1

ACTIONS, Finish

Add

IF TYPE = Dummy3

Go (3) = 1

ELSEIF TYPE = Dummy4

Go (4) = 1

ELSE

STOP

ENDIF

IF TYPE = Dummy3

Headwhere2 = "Top Lane"

ELSEIF TYPE = Dummy4

Headwhere2 = "Bottom Lane"

ENDIF

09873021.060101

End Actions

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: IF ISTATE (Machine002(1)) = 3 AND Go (3) = 0

PULL from Dummy3 out of WORLD

ELSE

Wait

ENDIF;

Output Rule: PUSH to SHIP;

Output_From: Front;

Output_Quantity: 0;

Cycle number: 2

* Cycle time: 0.01;

* Input quantity: 0;

* Finish quantity: 1;

* Description: Cycle2

LABOR:

Cycle: None;

Pre-empt level: None;

END

Input rule: Wait;

Output Rule: IF Go (3) = 0

PUSH to SHIP

ELSE

Wait

ENDIF;

Output_From: Front;

END CYCLE_DETAIL

BREAKDOWN_DETAIL

Breakdown number: 1

09873021.060101

* Mode: Busy time;
* Down interval: Undefined;
* Repair time: Undefined;
* Scrap part: No;
* Description: MTBF
* Setup on repair: No;
* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 0;

ACTIONS, Breakdown

Add

BREAKDWN (Conveyor001(1))

BREAKDWN (Machine001(1))

BREAKDWN (Conveyor002(1))

BREAKDWN (Machine002(1))

BREAKDWN (Head)

End Actions

ACTIONS, Repair

Add

REPAIR (Conveyor001(1))

REPAIR (Machine001(1))

REPAIR (Conveyor002(1))

REPAIR (Machine002(1))

REPAIR (Head)

End Actions

LABOR:

Repair: None;

END

Breakdown number: 2

* Mode: Busy time;
* Down interval: NORMAL (3600.0,500,48);
* Repair time: 3.5 * 60;
* Scrap part: No;

09873021.060101

* Description: MTBA
 * Setup on repair: No;
 * Check at cycle start only: Yes;
 DIALOG DISPLAY NUMBER: 1;
 ACTIONS, Breakdown
 Add

BREAKDWN (Machine002(1))

End Actions
 ACTIONS, Repair
 Add

REPAIR (Machine002(1))

End Actions
 LABOR:

Repair: None;

END

Breakdown number: 3

* Mode: Busy time;

* Down interval: TRIANGLE (2.5 * 1440 * 60,3 * 1440 * 60,3.5 * 1440 * 60,72);

* Repair time: 120.0;

* Scrap part: No;

* Description: Nozzle ID Error

* Setup on repair: No;

* Check at cycle start only: Yes;

DIALOG DISPLAY NUMBER: 2;

LABOR:

Repair: None;

END

END BREAKDOWN_DETAIL

REPORTING: Individual;

SHIFT: Undefined,0,0;

09873021.060101

2000P07660US01

END Head2

Conveyor003

NAME OF CONVEYOR: Conveyor003;

QUANTITY: 1;

TYPE: Queuing;

PART LENGTH: 1;

MAX CAPACITY: 1;

ACTIONS, Front

Add

STAGE = STAGE + 1

End Actions

INPUT RULE: Wait;

OUTPUT RULE: Wait;

CYCLE TIME: 2.0;

BREAKDOWNS: No;

PRIORITY: Lowest;

LABOR:

Repair: None;

END

REPORTING: Individual;

SHIFT: Undefined;

RESTART DELAY: Undefined;

END Conveyor003

Dummy1

NAME OF PART: Dummy1;

TYPE: Variable attributes;

GROUP NUMBER: 1;

09873021.060101

MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy1

Dummy2

NAME OF PART: Dummy2;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy2

Dummy3

NAME OF PART: Dummy3;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;

09873021:060101

PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy3

Dummy4

NAME OF PART: Dummy4;
TYPE: Variable attributes;
GROUP NUMBER: 1;
MAXIMUM ARRIVALS: 0;
OUTPUT RULE: Wait;
PART ROUTE: None
REPORTING: Yes;
CONTAINS FLUIDS: No;
SHIFT: Undefined;
ARRIVAL_PROFILE: No;

END Dummy4

Headwhere1

NAME OF VARIABLE: Headwhere1;
QUANTITY: 1;
REPORTING: Yes;

END Headwhere1

Headwhere2

09873021-060101

NAME OF VARIABLE: Headwhere2;

QUANTITY: 1;

REPORTING: Yes;

END Headwhere2

Pie001

NAME OF PIE_CHART: Pie001;

QUANTITY: 1;

STATE_VALUES:

* Name: Head;

* Type: standard;

* Report by: on_shift_time;

* Values: group;

REFRESH INTERVAL: 3600;

REPORTING: Individual;

PULL_OUT_SEGMENT: 0;

SHIFT: Undefined;

END Pie001

Pie002

NAME OF PIE_CHART: Pie002;

QUANTITY: 1;

STATE_VALUES:

* Name: Head2;

* Type: standard;

* Report by: on_shift_time;

* Values: group;

09873021.060101

2000P07660US01

REFRESH INTERVAL: 3600;
REPORTING: Individual;
PULL_OUT_SEGMENT: 0;
SHIFT: Undefined;

END Pie002

END_MODULE_DETAIL

END HS52

Lift1

NAME OF MACHINE: Lift1;
QUANTITY: 1;
TYPE: Single;
PRIORITY: Lowest;
LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: Lifftime;

INPUT RULE: Wait;

OUTPUT RULE: Wait;

Output_From: Front;

REPORTING: Individual;

SHIFT: Undefined,0,0;

09873021-060101

END Lift1

Lift2

NAME OF MACHINE: Lift2;

QUANTITY: 1;

TYPE: Single;

PRIORITY: Lowest;

LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: Liftime;

INPUT RULE: ! Top to bottom done

IF NPARTS (HS50.Conveyor003) > 0 AND NPARTS
(Underconvl(9)) = 0

PULL from HS50.Conveyor003 at Front

!

!! Bottom not done to next machine or next not done conveyor (not)

!

ELSEIF NPARTS (Underconvl(2)) > 0 AND (NPARTS
(HS51.Conveyor001) = 0 OR NPARTS (Underconvl(3)) = 0)

PULL from Underconvl(2) at Front

ELSE

Wait

ENDIF;

09873021-050101

OUTPUT RULE: Wait;
Output_From: Front;
REPORTING: Individual;
SHIFT: Undefined,0,0;

END Lift2

Lift3

NAME OF MACHINE: Lift3;
QUANTITY: 1;
TYPE: Single;
PRIORITY: Lowest;
LABOR:

Cycle: None;

END

DISCRETE LINKS :

Fill: None

END

DISCRETE LINKS :

Empty: None

END

CYCLE TIME: Lifftime;

INPUT RULE: ! Top to bottom done

IF NPARTS (HS51.Conveyor003) > 0 AND NPARTS
(Underconvl(11)) = 0

PULL from HS51.Conveyor003 at Front

!

!! Bottom not done to next machine or next not done conveyor (not)

!

ELSEIF NPARTS (Underconvl(4)) > 0 AND NPARTS
(HS52.Conveyor001) = 0 !or nparts(underconvl(3)))

PULL from Underconvl(4) at Front
 ELSEIF NPARTS (Underconvl(10)) > 0 AND NPARTS
 (Underconvl(11)) = 0
 PULL from Underconvl(10) at Front
 ELSE
 Wait
 ENDIF;
 OUTPUT RULE: Wait;
 Output_From: Front;
 REPORTING: Individual;
 SHIFT: Undefined,0,0;

 END Lift3

 Underconvl

 NAME OF CONVEYOR: Underconvl;
 QUANTITY: 12;
 TYPE: Queuing;
 PART LENGTH: 1;
 MAX CAPACITY: 1;
 INPUT RULE: Wait;
 OUTPUT RULE: Wait;
 CYCLE TIME: 1.0;
 BREAKDOWNS: No;
 PRIORITY: Lowest;
 LABOR:
 Repair: None;
 END
 REPORTING: Individual;
 SHIFT: Undefined;
 RESTART DELAY: Undefined;

09873021.050101

END Underconvl

Lifftime

NAME OF VARIABLE: Lifftime;

QUANTITY: 1;

REPORTING: Yes;

END Lifftime

END_MODULE_DETAIL

END HS50Cluster310101

END SELECT

END DETAIL

* * * END TEMPLATE * * *

The templates according to the present invention may contain certain values for the parameters for each piece of equipment or process step that are inherent and cannot be readily varied. For example, the conveyor speed, machine routing logic, and station to station time of a pick and place machine is often fixed. Some of these values may be proprietary. One advantage of burying the proprietary values in the template is that the contents of the template is usually not readily accessible to the user.

The template may also contain place holders for values for other parameters that may vary. These values correspond to the parameters and data in the populated spreadsheet. For example, Figure 7A, at Row 40, depicts certain parameters for a cluster of two Siemens® HS 50, such as Mean Time Between Failure, etc. The Simulation object template for the HS

50 cluster has corresponding parameters that utilize the actual values from the populated spreadsheets.

In order to build the simulation objects that are needed to run the simulation software, a simulation development tool interfaces the spreadsheet with the discrete simulation software. The development tool extracts data from the user interface spreadsheets and other sheets, e.g. board data, shift data, etc. It puts this data in a Transfer File, such as, CBM_Transfer.XLS, where the data can be used to build large and data intensive simulation objects in the discrete event simulation software. In one embodiment, the development tool uses an OLE connection to interface a spreadsheet, such as MS Excel®, with a discrete event simulation software package, such as Witness® 2000. Other simulation software, such as ProModel® (available from Promodel Corp., Provo Utah), Automod® (available from Autosimulations, Provo Utah), Arena® (available from Systems Modeling, Pittsburgh Pa), Saimple® (available from AESOP GmbH, Germany) can also be used. Preferably, the development tool has grouping functionality similar to that found in products such as Microsoft Powerpoint®. To avoid accidentally moving objects, a lock function can be incorporated. The lock can be on a module base and the lock may be password protected.

Below is exemplary code for a software tool that, as is shown as step 4 of Figure 1, extracts designer objects from an Excel Spreadsheet, builds simulation objects and places the current data in a Witness® 2000 simulation.

Software Tool

```

*      *      *      Begin Code for Software Tool!      *      *      *
|*****
|*****
| This section reads all of the data from the CBM_TRANSFER spreadsheet
| into WITNESS tables
|*****
|*****
|
ok = XLReadArray (stringpath +

```

```

"\lcbm_transfer.xls","tr_line","$B$1:$B$101",Siplacenumbers)
PRINT "siplacenumbers",ok
ok = XLReadArray (stringpath + "\lcbm_transfer.xls"
,"tr_line","$D$1:$D$100",Siplacepcnnums)
PRINT "siplacepcnnumbers",ok
ok = XLReadArray (stringpath + "\lcbm_transfer.xls"
,"lookup","$A$1:$B$43",routeadd)
PRINT "routeadd",ok
ok = XLReadArray (stringpath + "\lcbm_transfer.xls"
,"tr_brd_ct","$A$1:$A$100",cycletimes)
PRINT "cycletimes",ok
ok = XLReadArray (stringpath + "\lcbm_transfer.xls"
,"tr_line","$Y$1:$Y$100",lastx)
PRINT "lastx",ok
ok = XLReadArray (stringpath + "\lcbm_transfer.xls"
,"tr_line","$G$1:$H$101",Assistsdata)
PRINT "assists",ok
ok = XLReadArray (stringpath + "\lcbm_transfer.xls","tr_line","$M$1:$N$101"
,Failuresdata)
PRINT "failures",ok
ok = XLReadArray (stringpath + "\lcbm_transfer.xls"
,"tr_line","$E$1:$E$101",Singleordual)
PRINT "singleordual",ok

```

!

```

| *****
| *****

```

! This section relates the numbers from the front end to the WITNESS elements sitting in the designer elements box. Object

! chosen also depends on whether it is single or dual lane on the spreadsheet.

```

| *****
| *****

```

```

!
mccount = 0
FOR count = 1 TO 100
    mccount = mccount + 1
    IF Singleordual (count) = 1
        IF Siplacenumbers (count) = 1200 OR Siplacenumbers (count)
= 1230 OR Siplacenumbers (count) = 1250 OR Siplacenumbers (count) =
1040 OR Siplacenumbers (count) = 1050
            elname (mccount) = S25_L1
        ELSEIF Siplacenumbers (count) = 1042 OR Siplacenumbers
(count) = 1052
            elname (mccount) = S25Cluster2_L1
        ELSEIF Siplacenumbers (count) = 1043 OR Siplacenumbers
(count) = 1053
            elname (mccount) = S25Cluster3_L1
        ELSEIF Siplacenumbers (count) = 1500
            elname (mccount) = HS50_L1
        ELSEIF Siplacenumbers (count) = 1502
            elname (mccount) = HS50Cluster2_L1
        ELSEIF Siplacenumbers (count) = 1503
            elname (mccount) = HS50Cluster3_L1
        ELSEIF Siplacenumbers (count) = 1040
            elname (mccount) = S25_L1
        ELSEIF Siplacenumbers (count) = 1050
            elname (mccount) = S25_L1
        ELSEIF Siplacenumbers (count) = 7500
            elname (mccount) = Lastlift
        ELSEIF Siplacenumbers (count) = 9001 OR Siplacenumbers
(count) = 9002 OR Siplacenumbers (count) = 9003 OR Siplacenumbers
(count) = 9004
            elname (mccount) = Printer
        ELSEIF Siplacenumbers (count) = 8001 OR Siplacenumbers

```

09873021.060101

```

(count) = 8002 OR Siplacenumbers (count) = 8003 OR Siplacenumbers
(count) = 8004 OR Siplacenumbers (count) = 8005 OR Siplacenumbers
(count) = 8006

        elname (mccount) = Oven_1Lane
ELSEIF Siplacenumbers (count) = 7000
        elname (mccount) = Conv5_1
ENDIF
ELSEIF Singleordual (count) = 2
        IF Siplacenumbers (count) = 1200 OR Siplacenumbers (count)
= 1230 OR Siplacenumbers (count) = 1250 OR Siplacenumbers (count) =
1040 OR Siplacenumbers (count) = 1050
                elname (mccount) = S25
        ELSEIF Siplacenumbers (count) = 1042 OR Siplacenumbers
(count) = 1052
                elname (mccount) = S25Cluster2
        ELSEIF Siplacenumbers (count) = 1043 OR Siplacenumbers
(count) = 1053
                elname (mccount) = S25Cluster3
        ELSEIF Siplacenumbers (count) = 1500
                elname (mccount) = HS50
        ELSEIF Siplacenumbers (count) = 1502
                elname (mccount) = HS50Cluster2
        ELSEIF Siplacenumbers (count) = 1503
                elname (mccount) = HS50Cluster3
        ELSEIF Siplacenumbers (count) = 1040
                elname (mccount) = S25
        ELSEIF Siplacenumbers (count) = 1050
                elname (mccount) = S25
        ELSEIF Siplacenumbers (count) = 7500
                elname (mccount) = Lastlift
        ELSEIF Siplacenumbers (count) = 9001 OR Siplacenumbers
(count) = 9002 OR Siplacenumbers (count) = 9003 OR Siplacenumbers

```

09873021.060101

```
(count) = 9004
```

```
ename (mccount) = Printer
```

```
ELSEIF Siplacenumbers (count) = 8001 OR Siplacenumbers
(count) = 8002 OR Siplacenumbers (count) = 8003 OR Siplacenumbers
(count) = 8004 OR Siplacenumbers (count) = 8005 OR Siplacenumbers
(count) = 8006
```

```
ename (mccount) = Oven_2
```

```
ELSEIF Siplacenumbers (count) = 7000
```

```
ename (mccount) = Conv5_2
```

```
ENDIF
```

```
ELSEIF Singleordual (count) = 0
```

```
IF Siplacenumbers (count) = 7100
```

```
ename (mccount) = Shuttle1_2
```

```
ELSEIF Siplacenumbers (count) = 7200
```

```
ename (mccount) = Shuttle2_1
```

```
ELSEIF Siplacenumbers (count) = 7300
```

```
ename (mccount) = Buffers
```

```
ELSEIF Siplacenumbers (count) = 7500
```

```
ename (mccount) = Lastlift
```

```
ENDIF
```

```
ENDIF
```

```
NEXT
```

```
!
```

```
!
```

```
*****
```

```
*****
```

```
! This section sets the path to each of the modules
```

```
!
```

```
*****
```

```
*****
```

```
!
```

```
stringpathso = stringpathso + "\\\" + "\\\"
```

```

PRINT stringpathso
FOR X = 1 TO 100
!   IF elname (X) <> NAME2STR (Initial01)
      modname = STR2NAME (elname (X))
      ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n FILENAME: "
+ stringpathso + modname + ".MDL;\nEND" + modname + "\nEND
SELECT\nEND DETAIL")
      PRINT elname (X),ok
!   ENDIF
NEXT
!
!
!
*****
*****
! This section creates the actual WITNESS simulation elements
!*****
*****
!
FOR X = 1 TO 100
  IF elname (X) <> ""
      success = CreateFromDesigner (STR2NAME (elname
(X)),elname (X) + "0" + X,lastx (X),500,NONE,lastx (X) + 20,530)
      ellist (X) = STR2NAME (elname (X) + "0" + X)
  ELSE
      elname (X) = Initial01
      ellist (X) = Initial01
  ENDIF
NEXT
!
!
!*****

```

! This section establishes that the correct cycle time are inserted into each of the simulation elements.

!

!

mccount = 1

FOR count = 1 TO ELTCOUNT (1)

IF ELTTYPE (ELTNAME (count)) = 4 AND LEFTSTR (ELTNAME (count),3) = "buf"

mccount = mccount + 1

ELSEIF ELTTYPE (ELTNAME (count)) = 3 AND (LEFTSTR (ELTNAME (count),3) = "Con" OR LEFTSTR (ELTNAME (count),3) = "Shu")

mccount = mccount + 1

ELSEIF ELTTYPE (ELTNAME (count)) = 2 AND RIGHTSTR (ELTNAME (count),5) = "Lift4"

mccount = mccount + 1

!

ELSEIF ELTTYPE (ELTNAME (count)) = 3 AND LEFTSTR (ELTNAME (count),4) = "Oven"

strvar = "initial01.CycleTimes(" + mccount + ")/30"

periodpos = STRSTR (NAME2STR (ELTNAME (count)), ".")

modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME (count))), periodpos - 1))

ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nOven_2001\nCYCLE TIME: " + strvar + ";\nEND
Oven_2001\n END_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")

!

mccount = mccount + 1

!

09873021.060101


```

ELSEIF ELTTYPE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),9) = "Prntermc"
    strvar = "initial01.Cycletimes(" + mccount + ")"
    periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
    modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count)),periodpos - 1))
    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nPrntermc\n TYPE: Single;\n CYCLE TIME: " +
strvar + ";\nEND Prntermc\n END_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")
    mccount = mccount + 1
!
!
ELSEIF ELTTYPE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),4) = "Head"
    strvar = "initial01.Cycletimes(" + mccount + ")-0.01"
!
|*****1 MODULE
    periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
    modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count)),periodpos - 1))
    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nHead\n TYPE: General(Multi-Cycle);\n
CYCLE_DETAIL\n Cycle number: 1\n * Cycle time: " + strvar + ";\n *
Input quantity: 1;\n * Finish quantity: 1;\n * Description: Cycle number
1\nEND CYCLE_DETAIL\nEND Head\n END_MODULE_DETAIL\nEND " +
modname + "\nSELECT\nEND DETAIL")
!
|*****2 MODULES
    periodpos2 = STRLEN (NAME2STR (modname))
    modstr = MIDSTR (NAME2STR (ELTNAME (count)),periodpos2
+ 2,100)

```

09873021.060101

```

periodpos2 = STRSTR (modstr, ".")
modstr = LEFTSTR (modstr, periodpos2 - 1)
ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\n" + modstr + "\n
BEGIN_MODULE_DETAIL\nHead\n TYPE: General(Multi-Cycle);\n
CYCLE_DETAIL\n Cycle number: 1\n * Cycle time: " + strvar + ";\n *
Input quantity: 1;\n * Finish quantity: 1;\n * Description: Cycle number
1\nEND CYCLE_DETAIL\nEND Head\nEND_MODULE_DETAIL\nEND " +
modstr + "\n END_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")
!
mccount = mccount + 1
!
ELSEIF ELTTYPE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count), 4) = "ead2"
strvar = "initial01.Cycletimes(" + mccount + ")-0.01"
!
!*****1 MODULE
periodpos = STRSTR (NAME2STR (ELTNAME (count)), ".")
modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count)), periodpos - 1))
ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nHead2\n TYPE: General(Multi-Cycle);\n
CYCLE_DETAIL\n Cycle number: 1\n * Cycle time: " + strvar + ";\n *
Input quantity: 1;\n * Finish quantity: 1;\n * Description: Cycle number
1\nEND CYCLE_DETAIL\nEND Head2\n END_MODULE_DETAIL\nEND " +
modname + "\nSELECT\nEND DETAIL")
!
!*****2 MODULES
periodpos2 = STRLEN (NAME2STR (modname))
modstr = MIDSTR (NAME2STR (ELTNAME (count)), periodpos2
+ 2, 100)

```

09873021.060101

```

periodpos2 = STRSTR (modstr, ".")
modstr = LEFTSTR (modstr, periodpos2 - 1)
ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\n" + modstr + "\n
BEGIN_MODULE_DETAIL\nHead2\n TYPE: General(Multi-Cycle);\n
CYCLE_DETAIL\n  Cycle number: 1\n  * Cycle time: " + strvar + ";\n  *
Input quantity: 1;\n  * Finish quantity: 1;\n  * Description: Cycle number
1\nEND CYCLE_DETAIL\nEND Head2\nEND_MODULE_DETAIL\nEND " +
modstr + "\n END_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")
!
mccount = mccount + 1
!
ENDIF
NEXT
!
! *****
! This section forms the routing table for the boards
! *****
FOR count = 1 TO 100
  IF elname (count) <> ""
    X = IPOSSVAL (routeadd, elname (count), 2, 1)
    IPRINT count, X
    IF routeadd (2, X) <> "None"
      modoutput (count) = STR2NAME (ellist (count) + ". " +
routeadd (2, X) + "(N)")
    ELSE
      modoutput (count) = STR2NAME (ellist (count))
    ENDIF
  ENDIF
NEXT
!
```

```

|*****
! Shifts
|*****SETS  SHIFTS IN ALL MACHINES & CONVEYORS
|*****
READSHFT (Shiftpattern,stringpath + "\\tr_shft_.sft")
FOR count = 1 TO ELTCOUNT (1)
    IF ELTTYE ( ELTNAME (count)) = 2 OR ELTTYE (ELTNAME
(count)) = 3 OR ELTTYE (ELTNAME (count)) = 1 OR ELTTYE (ELTNAME
(count)) = 8
        SETSHIFT (ELTNAME (count),Shiftpattern)
    ENDIF
NEXT
|*****
|*****Assists*****
*
|*****
!RETURN
!
PRINT "V"
mccount = 1
FOR count = 1 TO ELTCOUNT (1)
    IF ELTTYE (ELTNAME (count)) = 4 AND LEFTSTR (ELTNAME
(count),3) = "bu"
        mccount = mccount + 1
    ELSEIF ELTTYE (ELTNAME (count)) = 3 AND (LEFTSTR
(ELTNAME (count),3) = "Con" OR LEFTSTR (ELTNAME (count),3) = "Shu")
        mccount = mccount + 1
    ELSEIF ELTTYE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),5) = "Lift4"
        mccount = mccount + 1
    !
    ELSEIF ELTTYE (ELTNAME (count)) = 3 AND LEFTSTR (ELTNAME

```

09873021.060101

```

--
(count),4) = "Oven"

      strvar = "Normal(initial01.Assists (1," + mccount + ") *2, 0.05 *
initial01.Assists (1," + mccount + "),"+ 9 + mccount + ")")
      strvar2 = "Normal(initial01.Assists (2," + mccount + ") , 0.05 *
initial01.Assists (2," + mccount + "),"+ 8 + mccount + ")")
      periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
      modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count))),periodpos - 1))

      ok = SETINFO ("DETAIL\SELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nOven_2001\nBREAKDOWNS: Available time;\n*
Down interval: " + strvar + "\n* Repair Time: " + strvar2 + ";\nEND
Oven_2001\n END_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")
!

      mccount = mccount + 1
!

      ELSEIF ELTTYPE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),9) = "Printermc"

      strvar = "Normal(initial01.Assists (1," + mccount + ") *2, 0.05 *
initial01.Assists (1," + mccount + "),"+ 9 + mccount + ")")
      strvar2 = "Normal(initial01.Assists (2," + mccount + ") , 0.05 *
initial01.Assists (2," + mccount + "),"+ 8 + mccount + ")")
      periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
      modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count))),periodpos - 1))

      ok = SETINFO ("DETAIL\SELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nPrintermc\nBREAKDOWN_DETAIL\nBreakdown
number: 2\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBA\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
BREAKDOWN_DETAIL\nEND Printermc\nEND_MODULE_DETAIL\nEND " +
modname + "\nSELECT\nEND DETAIL")

```

09873021.060101

```

mccount = mccount + 1

ELSEIF ELTTYE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),4) = "Head"

    strvar = "Normal(initial01.Assists (1," + mccount + ") *2, 0.05 *
initial01.Assists (1," + mccount + ")," + 9 + mccount + ")")

    strvar2 = "Normal(initial01.Assists (2," + mccount + ") , 0.05 *
initial01.Assists (2," + mccount + ")," + 8 + mccount + ")")

!*****1 Module

    periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
    modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count))),periodpos - 1))

    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nHead\nBREAKDOWN_DETAIL\nBreakdown
number: 2\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBA\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
BREAKDOWN_DETAIL\nEND Head\nEND_MODULE_DETAIL\nEND " +
modname + "\nSELECT\nEND DETAIL")

!*****2 Modules

    periodpos2 = STRLEN (NAME2STR (modname))
    modstr = MIDSTR (NAME2STR (ELTNAME (count)),periodpos2
+ 2,100)

    periodpos2 = STRSTR (modstr,".")
    modstr = LEFTSTR (modstr,periodpos2 - 1)

    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\n" + modstr + "\n
BEGIN_MODULE_DETAIL\nHead\n BREAKDOWN_DETAIL\nBreakdown
number: 2\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBA\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
BREAKDOWN_DETAIL\nEND Head\nEND_MODULE_DETAIL\nEND " +
modstr + "\n END_MODULE_DETAIL\nEND " + modname +

```

```
"\nSELECT\nEND DETAIL")
```

```
|
```

```
    mccount = mccount + 1
```

```
    ELSEIF ELTTYE (ELTNAME (count)) = 2 AND RIGHTSTR  
(ELTNAME (count),4) = "ead2"
```

```
        strvar = "Normal(initial01.Assists (1," + mccount + ") *2, 0.05 *  
initial01.Assists (1," + mccount + ")," + 9 + mccount + ")"
```

```
        strvar2 = "Normal(initial01.Assists (2," + mccount + ") , 0.05 *  
initial01.Assists (2," + mccount + ")," + 8 + mccount + ")"
```

```
!*****1 Module
```

```
    periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
```

```
    modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME  
(count)),periodpos - 1))
```

```
    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n  
BEGIN_MODULE_DETAIL\nHead2\nBREAKDOWN_DETAIL\nBreakdown  
number: 2\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair  
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBA\nSetup on  
repair: No;\n* Check at cycle start only: Yes;\nEND  
BREAKDOWN_DETAIL\nEND Head2\nEND_MODULE_DETAIL\nEND " +  
modname + "\nSELECT\nEND DETAIL")
```

```
!*****2 Modules
```

```
    periodpos2 = STRLEN (NAME2STR (modname))
```

```
    modstr = MIDSTR (NAME2STR (ELTNAME (count)),periodpos2  
+ 2,100)
```

```
    periodpos2 = STRSTR (modstr,".")
```

```
    modstr = LEFTSTR (modstr,periodpos2 - 1)
```

```
    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n  
BEGIN_MODULE_DETAIL\n" + modstr + "\n  
BEGIN_MODULE_DETAIL\nHead2\n BREAKDOWN_DETAIL\nBreakdown  
number: 2\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair  
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBA\nSetup on  
repair: No;\n* Check at cycle start only: Yes;\nEND
```

```

BREAKDOWN_DETAIL\nEND Head2\nEND_MODULE_DETAIL\nEND " +
modstr + "\n END_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")

```

```
!
```

```
    mccount = mccount + 1
```

```
ENDIF
```

```
NEXT
```

```
!
```

```
rowcount = 1
```

```
FOR count = 1 TO mccount
```

```
    Assists (1,count) = Assistsdata (1,rowcount)
```

```
    Assists (2,count) = Assistsdata (2,rowcount)
```

```
    Siplacepcnums (rowcount) = Siplacepcnums (rowcount) - 1
```

```
    IF Siplacepcnums (rowcount) <= 0
```

```
        rowcount = rowcount + 1
```

```
    ENDIF
```

```
NEXT
```

```
!*****
```

```
!Failures at Level 1
```

```
!*****
```

```
PRINT '\f'
```

```
mccount = 1
```

```
FOR count = 1 TO ELTCOUNT (1)
```

```
    IF ELTTYE ( ELTNAME (count)) = 4 AND LEFTSTR (ELTNAME
(count),3) = "buf"
```

```
        mccount = mccount + 1
```

```
    ELSEIF ELTTYE (ELTNAME (count)) = 3 AND (LEFTSTR
(ELTNAME (count),3) = "Con" OR LEFTSTR (ELTNAME (count),3) = "Shu")
```

```
        mccount = mccount + 1
```

```
    ELSEIF ELTTYE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),5) = "Lift4"
```

```
        mccount = mccount + 1
```


!

```
ELSEIF ELTTYPE (ELTNAME (count)) = 3 AND LEFTSTR (ELTNAME
(count),4) = "Oven"
```

```
    strvar = "Normal(initial01.failures1 (1," + mccount + ") *2, 0.05 *
initial01.failures1 (1," + mccount + "),"+ 3 + mccount + ")"
```

```
    strvar2 = "Normal(initial01.failures1 (2," + mccount + ") , 0.05 *
initial01.failures1 (2," + mccount + "),"+ 4 + mccount + ")"
```

```
    periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
```

```
    modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count))),periodpos - 1))
```

```
    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nOven_2001\nBREAKDOWNS: Available time;\n*
Down interval: " + strvar + "\n* Repair Time: " + strvar2 + ";\nEND
Oven_2001\nEND_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")
```

!

```
    mccount = mccount + 1
```

!

```
ELSEIF ELTTYPE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),9) = "Prntermc"
```

```
    strvar = "Normal(initial01.failures1 (1," + mccount + ") *2, 0.05 *
initial01.failures1 (1," + mccount + "),"+ 9 + mccount + ")"
```

```
    strvar2 = "Normal(initial01.failures1 (2," + mccount + ") , 0.05 *
initial01.failures1 (2," + mccount + "),"+ 8 + mccount + ")"
```

```
    periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
```

```
    modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count))),periodpos - 1))
```

```
    ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nPrntermc\nBREAKDOWN_DETAIL\nBreakdown
number: 1\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBF\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
```

09873021.060101

```
BREAKDOWN_DETAIL\nEND Printermod\nEND_MODULE_DETAIL\nEND " +
modname + "\nSELECT\nEND DETAIL")
```

```
mccount = mccount + 1
```

```
ELSEIF ELTTYPE (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),4) = "Head"
```

```
strvar = "Normal(initial01.failures1 (1," + mccount + ") *2, 0.05 *
initial01.failures1 (1," + mccount + ")," + 9 + mccount + ")"
```

```
strvar2 = "Normal(initial01.failures1 (2," + mccount + ") , 0.05 *
initial01.failures1 (2," + mccount + ")," + 8 + mccount + ")"
```

```
!*****1 Module
```

```
periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
```

```
modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count))),periodpos - 1))
```

```
ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nHead\nBREAKDOWN_DETAIL\nBreakdown
number: 1\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBF\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
BREAKDOWN_DETAIL\nEND Head\nEND_MODULE_DETAIL\nEND " +
modname + "\nSELECT\nEND DETAIL")
```

```
!*****2 Modules
```

```
periodpos2 = STRLEN (NAME2STR (modname))
```

```
modstr = MIDSTR (NAME2STR (ELTNAME (count))),periodpos2
+ 2,100)
```

```
periodpos2 = STRSTR (modstr,".")
```

```
modstr = LEFTSTR (modstr,periodpos2 - 1)
```

```
ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\n" + modstr + "\n
BEGIN_MODULE_DETAIL\nHead\n BREAKDOWN_DETAIL\nBreakdown
number: 1\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBF\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
```

```

BREAKDOWN_DETAIL\END Head\END_MODULE_DETAIL\END " +
modstr + "\n END_MODULE_DETAIL\END " + modname +
"\nSELECT\END DETAIL")
!
        mccount = mccount + 1
        ELSEIF ELTTY (ELTNAME (count)) = 2 AND RIGHTSTR
(ELTNAME (count),4) = "ead2"
                strvar = "Normal(initial01.failures1 (1," + mccount + ") *2, 0.05 *
initial01.failures1 (1," + mccount + "), + 9 + mccount + ") "
                strvar2 = "Normal(initial01.failures1 (2," + mccount + ") , 0.05 *
initial01.failures1 (2," + mccount + "), + 8 + mccount + ") "
!*****1 Module
                periodpos = STRSTR (NAME2STR (ELTNAME (count)),".")
                modname = STR2NAME (LEFTSTR (NAME2STR (ELTNAME
(count)),periodpos - 1))
                ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\nHead2\nBREAKDOWN_DETAIL\nBreakdown
number: 1\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair
time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBF\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
BREAKDOWN_DETAIL\END Head2\nEND_MODULE_DETAIL\END " +
modname + "\nSELECT\END DETAIL")
!*****2 Modules
                periodpos2 = STRLEN (NAME2STR (modname))
                modstr = MIDSTR (NAME2STR (ELTNAME (count)),periodpos2
+ 2,100)
                periodpos2 = STRSTR (modstr,".")
                modstr = LEFTSTR (modstr,periodpos2 - 1)
                ok = SETINFO ("DETAIL\nSELECT\n" + modname + "\n
BEGIN_MODULE_DETAIL\n" + modstr + "\n
BEGIN_MODULE_DETAIL\nHead2\n BREAKDOWN_DETAIL\nBreakdown
number: 1\n* Mode: Busy Time;\n* Down interval: " + strvar + ";\n* Repair

```

09873021.060101

```

time: " + strvar2 + ";\n* Scrap part: No;\n* Description: MTBF\nSetup on
repair: No;\n* Check at cycle start only: Yes;\nEND
BREAKDOWN_DETAIL\nEND Head2\nEND_MODULE_DETAIL\nEND " +
modstr + "\n END_MODULE_DETAIL\nEND " + modname +
"\nSELECT\nEND DETAIL")
!
```

```

mccount = mccount + 1
```

```

ENDIF
```

```

NEXT
```

```

!
```

```

ok = XLReadArray (stringpath +
"\lcbm_transfer.xls","tr_line","$D$1:$D$100",Siplacepcnums)
```

```

PRINT "siplacepcnumbers",ok
```

```

rowcount = 1
```

```

FOR count = 1 TO mccount
```

```

    Failures1 (1,count) = Failuresdata (1,rowcount)
```

```

    Failures1 (2,count) = Failuresdata (2,rowcount)
```

```

    Siplacepcnums (rowcount) = Siplacepcnums (rowcount) - 1
```

```

    IF Siplacepcnums (rowcount) <= 0
```

```

        rowcount = rowcount + 1
```

```

    ENDIF
```

```

NEXT
```

* * * END CODE FOR SOFTWARE TOOL * * *

After the simulation objects are built, they may be outputted, e.g., to the screen, in a spreadsheet form as is shown in Figure 9. A simulation may be run by selecting a button to execute the simulation, or the simulation may be programmed to run automatically after it is built.

After the simulation is run, outputs from the simulation software may be exported to an output spread sheet. When used to perform electronic assembly solutions, outputs may comprise data such as the number of circuit boards processed per time period of simulation and utilization of machines

and conveyors. The present invention could be used in other areas besides the electronic assembly area and would provide data appropriate to these areas.

The output spreadsheet may be configured to accept certain inputs, such as boundary conditions, exchange rates, and depreciation rates. In some embodiments, the output spreadsheet may contain buttons such as those shown in Figure 10. The spreadsheet may also be configured to provide useful machine and assemble line data, such as cost of ownership and machine statistics.

Selecting the "Boundaries" button opens the interface shown in Figure 11 and allows the user to enter boundary data. Selecting the "Setup" button results in the launching of an interface as depicted in Figure 12 and allows the user to enter Setup information. The user may also select the "Quality" button to launch an interface as is depicted in Figure 13. This allows the user to input quality data and criteria. Likewise, the user can input maintenance data by clicking on the "Maintenance" button, which launches another input interface permitting the entry of maintenance-related information. (See Fig. 14).

Figure 15 depicts a spreadsheet configured to provide cost of ownership data. The data may also be depicted graphically as is shown in Figure 16. Typical cost of ownership calculations may employ the following equations.

Cost per Year = Investment * depreciation(%)*(1+ Interest Rate) + Labor cost Operator + Labor cost Technician + Electricity cost + Air costs + Nitrogen costs + Floorspace costs + Spare part costs + Setup cost + Maintenance cost. All costs may be summarized per year.

Output per year = (Yearly working hours - setup time - maintenance time - quality time) * Real placement capacity (by simulation)

Quality time = time which is necessary to reproduce bad boards.

Figure 16A is a flowchart illustrating how a cost of ownership function may be configured. Typically, cost of ownership will be calculated in

conformance with SEMI E 35 standards, which are hereby incorporated by reference.

Central to a successful modeling and simulation effort is the arrival at an appropriate level of abstraction for modeling the system of interest. While simulations may be run at various levels of abstraction, tradeoffs between accuracy and speed often exist. One wants to represent the system with sufficient fidelity and detail to generate results that satisfy the developer's objectives. At the same time, this goal needs to be achieved without excessive use of scarce resources, which, among other things, include the time available to set up particular models of systems to be simulated, as well as the time for the simulations to execute.

According to the present invention, satisfactory results in the electronic assembly solutions area may be obtained by modeling assembly system configurations at a material flow level of abstraction. Other levels of abstraction may be used might be used consistent with other aspects of the present invention; however, a higher level of abstraction may produce less accurate results and a lower level of abstraction may require more resources and may give more detail than is necessary for adequate customer benefit modeling. For example, to simulate at a lower level of abstraction, more details may be required for the simulation objects.

Because the present invention allows simulation objects to be built, and simulations to be run, in a comparatively short period of time, it is possible to quickly and easily model actual production line configurations. This modeling allows the accuracy of the simulation to be accessed and also allows simulation parameters to be adjusted to improve the overall performance of the simulation tool. For example, if the cycle time of a particular machine were reduced by one second, the tool can be used to predict the effect on the entire line's performance.

The present invention provides numerous advantages when modeling and designing assembly lines and other systems. In particular, the present invention provides an easy-to-use customer benefit modeling tool that is readily adaptable for use in the electronic assembly systems equipment

and/or solutions areas. It enables a method for applying customer benefit modeling to permit new pricing and financing arrangements for the electronic assembly systems, equipment and solutions areas, and it provides for a consultative sales process that employs these modeling approaches. For example, consultants and others may employ the customer benefit modeling to develop performance-based contracts for configuring and/or selling of assembly and/or production systems. By incorporating a customer solutions database that employs standard calculations, all consultants employing the tools and techniques of the present invention will have at their disposal standardized calculating schemes. This will enable improved solutions business models based on analysis, modeling, simulation, and quantification of customer benefits.

The tools and techniques of the present invention may be used to provide a model-based, closed loop process control approach to electronic assembly. An optimizer module based on advanced computing methodology may be added, as well as a combined business rule repository/optimizer module for electronic assembly systems, equipment and/or solutions. A combination data mining module and business rule repository may also be added.

The present invention may also be enabled to function over a network, such as the Internet. Consultants and customers can then access the tools and methods of the present invention from virtually anywhere. In addition, a web-based version allows the simulation objects to be centrally maintained and the database of parameters for the designer elements may also be centrally maintained and readily updated.

The present invention is particularly well suited for use by consultants wishing to improve the efficiency of assembly lines. Figure 17 illustrates the logic flow of how the present invention may be used. A consultant first inputs, in a defined user interface, a configuration resembling an existing assembly line, step 1001. The consultant may do this by selecting or inputting predefined Designer Elements, 1002, which may include values for certain operating parameters associated with the equipment and line being modeled.

These values may be stored in a separate database, which can be periodically updated. This results in a base case configuration, Step 1003. Next, various data, such as actual customer data inputs, 1004, including actual operating parameters from the existing line may be incorporated and used to adapt the default data of the base case, step 1005. If data on the actual operating parameters is not available, default data can be used. Next, a simulation is re-run using the actual customer data, step 1007. The values obtained by the model can be compared with the customer's actual production line data

Next, at step 1009, the consultant, using his own consulting knowledge, may attempt to optimize certain line parameters, such as cycle times, times to repair a particular machine, length or speed of conveyors, etc. It is also possible to allow the simulation software to automatically optimize some of these parameters. For example, Witness® 2000, contains a function to do this automatic optimization. Some of these parameter values may have initially been incorporated from the data associated with the Designer Elements.

Another simulation is created, step 1011. If the simulation produces erroneous results, it may be necessary to redefine and re-adapt the default data to the project at step 1005. If the results produced at step 1011 are reasonable, the configuration and associated parameters can be stored in a database, 1020, or the optimization step 1009, may be re-run to further optimize the line.

Logic flow line 1050 in Figure 17 shows one possible embodiment where the operating parameter database associated with the designer elements is updated based on real world experience. This database may be updated and be customized to reflect actual experiences of a particular customer.

The customer benefit modeling tool of the present invention may support various business models and methods. For example, the tool may be used as part of a method of doing business for an electronics assembly solutions business, such as is outlined in Figures 18 and 19. An electronics

assembly solutions business method may employ various phases, as is outlined in Figure 20. In phase 1, a PCB assembly line may be modeled statically and a simulation may be run using performance metrics that are estimated and may perform sensitivity analysis. In a phase 2, data may be collected from the PCB assembly line. In phase 3, data mining may occur through a Facilitator. The data collected may be used to perform modeling and simulation, as well as parameter estimation to validate performance metrics, perform sensitivity analysis, to perform forecasting and prediction. A business rule repository may be implemented that may, in some embodiments, acquire expert knowledge – from consultants and customers. The business rule repository may exchange information with modeling and simulation parameter groups, which may in turn with the business rules repository data be used to do line optimization. Performance may be optimized by the use of an Inference Engine. The method may be web-enabled using web portals and may provide feedback to the PCB assembly line for closed loop control.

The electronics assembly solutions business method may focus on the line, multi-line and shop floor levels in a customer's electronics assembly facilities. Customers may be both original equipment manufacturers ("OEMs") and contract manufacturers ("CMs"). A solutions business method may encompass a scope of supply that includes placement machines and other machines from Certified Partners, line level software products and a full range of service and support capabilities. The sales of electronics assembly solutions may be managed through a new Consultative Sales Process based on Customer Value Space Analysis and Fulfillment. The primary opportunities to increase customer solutions value include: Increased utilization, throughput, yield, flexibility and responsiveness; Reduced inventories; and Effective new product and new technology introduction. Long term performance may be optimized through bundled packages of innovative software and consulting services. This may enable real people in real factories to achieve high levels of ongoing system performance. Once a system has been modeled and implemented, real production data can be

used to calibrate the model and improve the accuracy of future system changes.

Figure 21 graphically represent the advantages that are achievable with the tools and methods of the present invention when compared with prior art tools and methods. As is graphically depicted, key enablers, i.e. products, methodologies, practices, tools, software applications, services, etc. enable customers to improve their performance and enhance performance metrics. Enhancement may be improved as part of an ongoing production line.

The flowchart in Figure 22 graphically illustrates a solutions value proposition from a customer project life cycle view. First, new line requirements are defined. Next, the basic configuration is selected. Suggested modifications are simulated, performance is analyzed, and the line configuration is optimized. A final line configuration is defined and a customer is shown the value of the optimization. Key performance metrics are defined and a contract implementing the optimization is negotiated.

Next, orders are placed for necessary equipment and components. The new line is then installed and commissioned. The line is monitored and performance problems are defined as part of a comprehensive performance analysis. An optimization plan is defined and the customer is shown the value of the optimization. A contract is then negotiated and required line changes are identified. The process may be repeated until the desired level of optimization is reached.

The customer benefit modeling tool of the present invention enables a business method that includes the steps of providing a specific set of documented, quantitative consulting process by trained consultants within a defined consulting framework. In an embodiment of the consulting process according to the present invention, a consultant: benchmarks existing customer operations; identifies and prioritizes solutions opportunities at a customer site; designs optimized solutions comprising machines, software tools, and operating parameters, such as staffing; defines achievable target performance metrics for proposed solutions; and quantifies and shows the customer benefit of implementing the proposed solutions.

The tools and methods of the present invention define: optimum shop floor configuration for a defined set of products and production volume; optimum line configurations and specific equipment within the shop floor; optimum batch sizing; set-up and changeover strategy; achievable target set-up and changeover metrics; achievable target run-time performance metrics; the operating environment needed to achieve target performance matrix, such as staffing, material supply, maintenance, operating rules; when and how to introduce specific new technology; and when and how to introduce new products.

The customer benefit modeling tool can accurately predict the performance of real production systems because it uses a dynamic model based on discrete event simulation. Accordingly, it enables customer system performance to be expressed in a range of metrics, including for example, throughput, cost performance, yield, OEE, and COO.

An embodiment of a method according to the present invention of using the customer benefit modeling tools described above involves managing a given customer's specific project through a series of phases defined by a consultative sales process, as is outline in Figure 23. These method steps include:

Step, 450, phase 1 – Identify opportunity. In this step, which is a solution level, custom details, primary issues and goal are identified. Key criteria, such as needs, willingness to pay, and potential costs and savings are also identified. This step may be performed by a consultant or salesperson. If performed by a salesperson, a consultant may provide any necessary support.

Step 460, phase 2 – Identify specific high level analysis. In this step, specific opportunities are identified for detailed analysis. The customer benefit tool of the present invention enables a consulting process, which may include showing a rough solution with value potential to a customer. In some embodiments, this will be generated on the fly during a sales call or initial meeting with the client. "What if" scenarios can be evaluated quickly to illustrate the value that can be added for the customer.

Step 470, phase 3 – Detailed Customer value analysis. In this step, detailed measurements and analysis for each area covered in phase 2 can be performed. The customer benefit modeling tool of the present invention enables a consultant to show accurately the solutions value potential. “What if” changes and benefits can be analyzed. In this phase, more detailed analysis is performed. In one embodiment, the consultant is paid for his time and efforts in this phase, whereas in phase 2, the consultant or sales person may make only sales pitches and therefore not be compensated for the effort expended there.

Step 480, phase 4 – Implementation plan and contract. In this phase, a detailed performance improvement plan for each area from phase 3 is developed. A value based implementation contract may be drawn-up and negotiated. Also in this phase, the equipment, software, implementation plan, scope of consulting and technical services can be specified.

Step 490, phase 5 – Implement. In this phase, the plan of phase 4 is implemented. The proposed customer benefits of the proposed solution are validated. This phase provides an opportunity for the consultant to demonstrate excellent implementation capability. The consultant also has the opportunity in this phase to become a preferred supplier (over internal resources) for the next project. The consultant, in some embodiments, is paid on a value added basis.

In an embodiment of the consultative sales process, all results from customer specific projects are used to build a customer solutions database. This database enables: Benchmarking of the current and proposed solutions; Proliferation of “best practice;” Consistency of processes and solutions; and Minimization of overall solution variety.

A consulting knowledge management process can be used to organize, update, and disseminate knowledge about the customer benefit tool, the consulting process, the consulting framework and the customer solutions database. In one embodiment, the consulting knowledge management process comprise: Documenting; Training; Performing case studies; and Meeting to discuss “Best Practices.”

The customer benefit modeling tool of the present invention is particularly well-suited to enable electronics assembly solutions selling based on customer value. The tool allows firm performance metrics to be established for any electronics assembly solutions project prior to implementation. The metrics may span the range from throughput, basic cost performance and yield, through overall equipment effectiveness ("OEE") and onto full cost of ownership ("COO"). The tool enables a consultant to accurately quantify unfulfilled value spaces at a customer site; design an optimized assembly solution to "fill the spaces" – this results in low cost of production and other customer specific goals; and to quantify and show the customer benefit of the proposed solution.

The customer benefit tool of the present invention enables unique and innovative sales and marketing techniques by offering low-risk line level performance guarantees (e.g. throughput, yield, OEE, COO, unit assembly cost). The present invention also offers gain share pricing solutions packages.

The tools and methods of the present invention may be used to model various processes and production environments. One primary target domain is for electronics assembly/test line and shop floor levels, although it is also well-suited for use in the general electronic assembly factory level, including sub-assembly levels of building, final assembly, testing, and packaging). In some key sectors (e.g. cellphone manufacture) electronics assembly and testing are directly in-line with final assembly and testing. In situations such as these, final assembly and testing may be considered part of the electronics assembly line for modeling purposes. Electronics assembly/test lines typically comprise a linked series of linked processing machines and board handling equipment. The handling equipment may include WIP buffers. Local, dedicated, component storage areas may be present. Individual machines may be able to buffer boards between one or more process stages and lince can be single or dual conveyor, synchronous or asynchronous operations.

The primary processes and environments that may be modeled may include SMT assembly (print, dispense, place, reflow /cure), through hole assembly (auto, manual, pin/connector insertion), inspection (manual, auto-optical, x-ray), and Test (in circuit, functional). An electronics assembly shop floor typically comprises: multiple lines that start and finish with a magazine load/unload unit; workcells containing groups of machines (not all in-line); WIP buffers; local component storage areas; tooling storage areas; and feeder set-up/kitting areas.

One business method of the present invention may employ the use of certified machine suppliers. All machines that are offered as part of an electronics assembly solutions will come from a certified OEM partner. Certification may require providing and maintaining for each of the specific equipment/option configurations: Tools for calculation nominal cycle time for a defined electronic assembly; Tools for calculating nominal process yield for a defined electronic assembly; Full and validated COO model, including accurate MTBF/MTTR data; Full classification/definition of interrupts that can decrease throughput from nominal levels; Information/acceptable roadmap for automated provision of key event data automatically available from equipment; and Full classification/definition of factors that can increase defects above nominal levels.

Assembly lines that may be the subject of the consultative sales process and that are suitable for use with the customer benefit model tool include:

Type A lines, which are very high volume, low mix lines having the following characteristic: 100 to 300k components per hour ("cph"); Up to 14 assembly systems per line (top and bottom assembly); Up to 40 total machines per line with a line length up to 100m; Fixed set-up (i.e. less than 1 product change per month); 24x7 production; and Pulse time 10 to 45 seconds.

Type B lines, which are High Volume, Medium Mix lines having the following characteristics: 40 to 120 cph; Up to 6 SIPLACE in line (single sided

assembly); Up to 25 machines per line with a line length up to 50m; Flexibility through "cluster configuration;" and 24x5 operation.

Type C lines, which are medium volume, high mix lines and have the following characteristics: 15 to 50 cph; Up to 3 SIPLACE machines per line; Up to 15 machines per line with a line length up to 25m; Small batch sizes, wide component and packaging type variety; and Ratio of production to set-up/changeover time can be less than 1:1.

A consultant may use the customer benefit modeling tool of the present invention to develop a basic line scenario. As is shown in Figure 24, the process flow for developing the scenario may start with the defining of a line process sequence. This may be selected from a library of supported process stages. Next the line equipment configuration is defined. This includes selecting models, number of units and options. These may be selected from a library of supported machines. In one embodiment, all have certified/expected COO models associated with them.

Next, in the "products defined" step, a process sequence is entered and for each stage in the process cycle time, composite yield, cost of material added, and handling data are inputted. This definitional step may be manual or may be automated through the use of such tools as Siemens SICALC tool. Next, a production schedule is defined. Set-up cycle time for a first cluster is entered, as is set-up and changeover time. This may be manually inputted or may be automatically inputted from a tool such as Siemens SIPLACE cluster tool. The process may require running several iterations to get a consensus on basic line design.

The customer benefit model tool of the present invention may also be used to perform customer project lifecycle development. This lifecycle development process is outline in the flowcharts of Figures 25 to 29. As shown in Figure 25 the process can line and system design. It can include optimized design capabilities (see Fig. 26); it may include automated performance monitoring. (See Fig. 27). In some embodiments data between design and monitoring activities may be shared. (See Fig. 28). A common

data base may also be employed to aid in performance optimization. (See Fig. 29).

Line Performance Monitoring/Basic Analysis.

One aspect of the present invention is that it allows for line performance monitoring and basic analysis. In one embodiment a line level software application that can be built upon is implemented. All machines in the line may be connected to together through an interface or in any suitable manner. The application monitors and records key time-based event data, such as "Board xyz is entering," or "Begin processing on Boardxyz," or "Board ID=xyz001 is complete," or "Part ABC123 is rejected at pickup location 15." Preferably, the application supports machine/line level performance reporting and simple "drill down" capabilities defined to current accepted industry standards, for example from Semiconductor Equipment and Materials International ("SEMI") or Association Connecting Electronic Industries ("IPC"). Basic line performance guarantees are fully enabled. This embodiment requires the installation of "Solution X" infrastructure (server, framework, database - at least one per site). In this embodiment connectivity to each machine is achieved with interface/adaptor kits (i.e. network adapters, drivers, etc.) Other software modules may "Plug and Play" into the infrastructure. This minimizes customer time and the costs involved.

This aspect of the invention is applicable to various manufacturers equipment. The lines that may be monitored may be proprietary, such as Siemens SIPLACE® lines, or they can be comprised of equipment from various competitors.

This aspect of the invention provides for a targeted sales business method where line monitoring capabilities are installed on all new lines. It also enables concurrent/subsequent sale of high value/high margin customer benefit modeling, consulting and sale of software packages.

The customer benefit modeling tool of the present invention may also be used to optimize line set-up and line change over. One method of optimization involves optimizing set-up and changeover strategy to achieve

required production goals. The set-up process can be managed to achieve a target performance.

The present invention provides for a consultative method that may include the steps of developing a strategy for optimization, developing an implementation plan, and showing the benefits of the plan to the customer by using the customer benefit modeling tool of the present invention.

Line Set-up and Changeover Optimization.

In one embodiment, the method of line set-up and changeover optimization may be implemented by software to optimize clustering, perform comprehensive set-up verification, perform changeover management, and handle local logistics.

Optimizing clustering involves consideration of comprehensive production constraints. These may include: board output requirements (wip, lot cycle time, due date); nominal changeover time; component placement rate; number of component reels (or trays, sticks); and number of feeders and feeder tables.

Comprehensive Set-up Verification includes external set-up; it is preferably performed in a closed loop fashion to avoid mistakes. It should cover critical parts, such as value, reworkability, test coverage, and shortages). Changeover Management involves the development of nominal schedules, monitoring key events, and configuring escalation processes/messages. Local Logistics often involve completing a list of required materials to ensure set-ups for the next time intervals.

The benefits of this method include equipment efficiency increases of up to 35%. Less scrap is generated and rework is reduced. This method also results in less set-up errors. The customers that may be prime targets for this method include all Type B and C production, and some Type A (e.g. prototype/pilot runs). When this method is employed, the customer benefit modeling tool may provide to the customer the impact of different clustering and set-up strategies on OEE/COO, the impact of effective changeover management on OEE/COO, the impact of set-up verification on OEE/COO. The cost of a single mistake and the material and rework cost elements may

also be shown to a customer. The customer can see the value of the consultants process in terms of real dollars because the customer benefit model can be quickly and easily run for the customer.

The customer benefit tool of the present invention also provides for a method for optimizing line performance during runtime. According to this method, a consultant designs an optimized line. The consultant may select a specific configuration or add or select specific equipment that will result in optimization (e.g. buffers and feeders). Staffing requirements are defined and may include (number, skill level, and tasks). Local material management practices/levels are also defined and may include components, WIP, and consumables. PM strategy is also defined and may include frequency and duration.

Software may be employed to perform line monitoring. It may incorporate additional events not provided in the basic monitoring package. The software may also be useful for performing analysis, such as providing a range of basic analysis relevant to different "User Views." In other words, each user, depending on his or her role, will have a different view of the information provided by the entire system. Accordingly, each role within an organization will have access to the right information/applications in the right form to effectively perform their tasks. For example, a line operator will have another set of information than the line supervisor or the maintenance technician or the production manager. As a further example, a solution may be expressed in terms of capacity for line management and terms of cost of ownership for executive management. The software may assist with local logistics. The software can manage, among other things, lists of required materials to avoid run out during the next time interval(s). It can also provide reports of separate scrap usage of components, as well as packaging related groups.

The benefits of the line runtime performance optimization method include the following. Equipment effectiveness may be increased by up to 35%. Bottleneck analysis and optimization may be provided. Material run out may be avoided. And other interrupts, such as feeder jams, mispicks, etc.

may be minimized. Scrap and rework costs are reduced. The method results in the elimination of incorrect components added. It also results in the process being optimized. Staffing costs are also optimized.

This method may be targeted to Type A and B production customers, as well as some Type C customers. The customer benefit tool of the present invention works well with this method to provide to the customer: the impact of different line configurations on OEE/COO; the impact of material run out on OEE/COO; the impact of incorrect material replenishment on OEE/COO; the Impact of Process Yields on OEE/COO; and the Impact of different staffing strategies on COO/OEE.

An embodiment of a consultative sales method according to the present invention is shown in flowchart form in Figure 30. This embodiment of the method, like those described below, may be implemented over any time frame, but may be most suitable for performing during a consultation or sales session with the customer. First, at 3002, the consultant or salesperson, after becoming familiar with the customer's electronics assembly system needs, identifies a corresponding set of one or more potential solutions opportunities. The solutions opportunities may involve any combination of equipment and parameter settings that may be involved in the design and/or configuration of an electronics assembly system. At 3004, the consultant or salesperson models the customer's electronics assembly system, which is preferably done in "real time," that is with the customer present, using the methods and tools described above. In this embodiment, as with the others described herein, this approach is done using a laptop computer on which all the necessary functionality for the methods according to the present invention have been programmed.

At 3006, one or more performance metrics may be defined for the electronics assembly system. This definition can occur at any time prior to evaluating the system, and could well be determined prior to the consulting or sales meeting by the customer. At step 3008, the potential solutions are prioritized by including them in the model and running the model under these configurations, and then by comparing them with each other and possibly with

09873021.060101

the one or more performance metrics. At step 3010, one or more proposed solutions may be selected from among those that have been prioritized, preferably, although not necessarily, according to their priority. At step 3012, the benefit of the proposed solution or solutions are quantified, again according to the methods described above. When the quantified benefit(s) have been computed, they can then be communicated to the customer for his or her consideration.

Referring to Figure 31, steps 3102 to 3112 are similar to the corresponding and like numbered steps of the flowchart in Figure 30. At step 3114, the configuration of the modeled assembly system under the one or more potential solutions may be modified, which could be the case, for example, if the results of the modeling of those potential solutions (as evidenced, e.g., in the performance metrics) did not compare favorably with customer needs and/or expectations. In this event, the benefit(s) of the modified proposal(s) are again quantified at 3116 (which would involve running the model again under the modifications), at which point a final quantified benefit may be presented to the customer. Steps 3114 and 3116 could be repeated as necessary, time permitting, to explore various configuration options, or to search for new possible solutions in the event that the predicted performance of the previous potential solutions have not approached customer needs.

Another embodiment of a consultative sales method according to the present invention is shown in Figure 32, in which the configuration, modeling and evaluation of electronics assembly configurations is performed in an iterative manner until a performance measure satisfies one or more customer constraints. A set of customer constraints, needs, or expectations are first identified, at 3202. As described above, this identification, and the rest of the method, can ideally be done according to the present invention, in real time (or "on the fly") during the course of a particular consultation or sales meeting with the customer. At 3204, an electronics assembly configuration (or solution) is selected. Then, at 3206, a model is established for the customer electronics assembly system comprising the selected configuration. The

model is run, at 3208, to generate at least one performance measure (which as in the embodiment of Figure 30 should have been arrived at in advance). The at least one performance measure is then compared, at 3210, against the customer constraints. If, as determined at 3212, the at least one performance measure does not satisfy customer constraints, then a new electronics assembly configuration or solution can be selected, at step 3204, and the process is repeated. If, however, the performance measures, as determined by the modeling process, do meet customer constraints, then at 3214, the consultant or salesperson may offer to sell to the customer at least a subset of the solution or of the electronics assembly equipment associated with that configuration or solution.

Turning to Figure 33, an alternative embodiment of a consultative sales method according to the present invention involves first establishing at 3302 a model of a customer's electronics assembly system, which can have a plurality of possible configurations or solutions. A measure of performance for the assembly system is selected, as in the descriptions presented above, in accordance with customer needs at step 3304, and a subset of the plurality of the configurations (which may be all the configurations) are selected for evaluation. At step 3308, a set of one or more criteria or selected for comparison against the subset of the evaluated configurations. The criteria could, alternatively, have been set well in advance of the method by the customer alone, or by the customer with the assistance of the consultant/salesperson, or, as shown, might theoretically be modified if appropriate during iterations of the approach. The model is then run, at 3310, to generate a predicted measure of performance for each of the subset of configurations selected at 3306 to be evaluated. The results are compared at 3312 to the criteria in order to select at least one preferred configuration. If, as determined at step 3314, it is determined that none of the configurations or solutions meets the customer criteria, control is returned to step 3306, where a new set of configurations or solutions may be selected or otherwise arrived at. If, on the other hand, an evaluated configuration or solution meets the customer's criteria, then the method is complete, but could be followed by

09873021.060101

any number of follow sales scenarios. As before, this consultative sales process is preferably implemented using the methods and tools described above in the course of a consultation or sales meeting with a customer.

Another aspect of the present invention concerns methods for pricing or financing electronics assembly solution or equipment purchases. This aspect of the present invention is based on the recognition of the potential value of customer benefit sharing in the electronics assembly industry, as well as in the practicability of such an approach given the methods and tools for modeling and evaluating such systems, as described above.

One customer benefit sharing approach according to the present invention is shown in the flowchart of Figure 34, which is directed to a method for pricing an electronics assembly system solution or configuration. A customer benefit that might be realized through the adoption of a proposed electronics assembly solution or configuration is predicted (e.g., using the methods and tools described above) at step 3402. A customer benefit computation may involve any measure that affects a customer's actual or perceived valuation for an electronics assembly system, including without limitation, the cost of ownership of the electronics assembly system, described above and in the literature. The purveyor of electronics assembly equipment, or of solutions involving such equipment, may then generate, at step 3406, a customer benefit guarantee associated with the electronics assembly solution. The purveyor's ability to generate such a guarantee with any comfort stems from its ability to effectively use the methods and tools for customer benefit modeling described above. When a customer benefit guarantee has been arrived at, it is then communicated to the customer at step 3406. The purveyor can then, at step 3408, offer the electronics assembly system configuration or solution to the customer at a price that is a function of the guaranteed customer benefit. For example, the price may be set as a mutually agreeable percentage of the guaranteed benefit, so that, from the customer's perspective, the electronics assembly configuration or solution is seen as "paying for itself" in real economic terms. In addition,

09873021.060101

consistent with the guarantee, the purveyor could bind itself to receive compensation, or full compensation, only if the guaranteed customer benefit were actually to be realized.

Figure 35 shows an alternative approach for a purveyor of electronics assembly equipment and/or solutions involving such equipment and/or related services, to engage in benefit sharing with a customer. At step 3502, an electronics assembly system solution or configuration is defined and the definition provided for customer consideration. A customer benefit associated with the provided solution or configuration is computed at 3504 above, and a monetary value based on the customer benefit is computed at 3506, both steps in accordance with the techniques described. Finally, an economic amount based on the computed monetary value is collected from the customer, at step 3508. The collectible amount could be, for example, a fraction of the monetary value of the customer benefit computed at step 3506, which fraction could be the subject of negotiation between the purveyor of electronics assembly equipment or solutions and the customer. Alternatively, the collectible amount could be a function of the customer benefit actually realized, which would involve monitoring the actual performance of the electronics assembly system or solution purchased by the customer.

Figure 36 provides a flowchart showing a method for selling a customer an electronic assembly system solution according to a performance-based contract. The purveyor of electronic assembly systems or solutions offers, at 3602, a performance based contract for a particular sale, where the price associated with the contract is set as a function of the performance of the system or solution that is being offered for sale. If the customer accepts the offer, as determined at 3604, the electronics assembly system solution (which may comprise assembly equipment), is delivered to the customer at 3606. If at 3604 the customer did not accept the offer, the method would be complete, although it could be re-initiated with a modified offer at 3602. Following the delivery at step 3606, the performance of the deliverables are monitored at step 3608. Based on the monitored performance, as well as on the price function set at 3602, the purveyor's

compensation - i.e., the price for the deliverables - is assessed. In this way, the benefit to the customer is shared between the customer and the purveyor. Actual payment could either be made in advance, with adjustments made as the performance-based assessments are made, or could be made following the assessments, depending upon the two parties' preferences. As with the previous examples, the purveyor's ability to comfortably enter into this type of transaction depends at least in part upon an ability to adequately model the expected customer benefit in advance, which can be done using the methods and tools described in this application.

In addition to the embodiments of the aspects of the present invention described above and in the code listings set forth herein, those of skill in the art will be able to arrive at a variety of other arrangements and steps which, if not explicitly described in this document, nevertheless embody the principles of the invention and fall within the scope of the appended claims.

09873021.060101